



Red Hat JBoss Enterprise Application Platform 7.3

Configuration Guide

Instructions for setting up and maintaining Red Hat JBoss Enterprise Application Platform, including running applications and services.

Red Hat JBoss Enterprise Application Platform 7.3 Configuration Guide

Instructions for setting up and maintaining Red Hat JBoss Enterprise Application Platform, including running applications and services.

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

The purpose of this guide is to cover many of the configuration tasks needed for setting up and maintaining JBoss EAP as well as running applications and other services on it.

Table of Contents

CHAPTER 1. INTRODUCTION	22
CHAPTER 2. STARTING AND STOPPING JBOSS EAP	23
2.1. STARTING JBOSS EAP	23
Start JBoss EAP as a Standalone Server	23
Start JBoss EAP in a Managed Domain	23
2.2. STOPPING JBOSS EAP	24
Stop an Interactive Instance of JBoss EAP	24
Stop a Background Instance of JBoss EAP	24
2.3. RUNNING JBOSS EAP IN ADMIN-ONLY MODE	24
Running a Standalone Server in Admin-only Mode	24
Start the Server in Admin-only Mode	24
Check If the Server is Running in Admin-only Mode	24
Reload in a Different Mode from the Management CLI	25
Running a Managed Domain in Admin-only Mode	25
Start a Host Controller in Admin-only Mode	25
Check If a Host Controller is Running in Admin-only Mode	25
Reload in a Different Mode from the Management CLI	25
2.4. SUSPEND AND SHUT DOWN JBOSS EAP GRACEFULLY	25
2.4.1. Suspend Servers	27
Check the Server Suspend State	27
Suspend	27
Resume	27
Start a Server in a Suspended State	28
2.4.2. Shut Down Servers Gracefully Using the Management CLI	28
2.4.3. Shut Down Servers Gracefully Using an OS Signal	29
2.5. STARTING AND STOPPING JBOSS EAP (RPM INSTALLATION)	29
2.5.1. Starting JBoss EAP (RPM Installation)	29
Start JBoss EAP as a Standalone Server (RPM Installation)	29
Start JBoss EAP in a Managed Domain (RPM Installation)	30
Configure RPM Service Properties	30
2.5.2. Stopping JBoss EAP (RPM Installation)	31
Stop JBoss EAP as a Standalone Server (RPM Installation)	31
Stop JBoss EAP in a Managed Domain (RPM Installation)	31
2.6. POWERSHELL SCRIPTS (WINDOWS SERVER)	32
CHAPTER 3. JBOSS EAP MANAGEMENT	33
3.1. ABOUT SUBSYSTEMS, EXTENSIONS, AND PROFILES	33
Using the Management Console or the Management CLI	33
3.2. MANAGEMENT USERS	33
3.2.1. Adding a Management User	34
3.2.2. Running the Add-User Utility Non-Interactively	34
Create a User Belonging to Multiple Groups	35
Specify an Alternative Properties File	35
3.2.3. Add-User Utility Password Restrictions	35
3.2.4. Updating a Management User	37
3.3. OPTIMIZING THE JBOSS EAP SERVER CONFIGURATION	38
3.4. MANAGEMENT INTERFACES	38
3.4.1. Management CLI	38
Launch the Management CLI	38
Connect to a Running Server	38

Display Help	39
Quit the Management CLI	39
View System Settings	39
Update System Settings	39
Start Servers	39
3.4.2. Management Console	39
3.4.2.1. Updating Attributes in the Management Console	40
3.4.2.2. Enable/Disable Management Console	40
3.4.2.3. Change the Language of the Management Console	41
To Change the Language of the Management Console	41
3.4.2.4. Customizing the Management Console Title	41
3.5. MANAGEMENT APIS	42
3.5.1. HTTP API	42
Read Resources	42
Update Resources	43
3.5.2. Native API	43
3.6. CONFIGURATION DATA	44
3.6.1. Standalone Server Configuration Files	44
3.6.2. Managed Domain Configuration Files	45
3.6.3. Backing Up Configuration Data	45
3.6.4. Configuration File Snapshots	46
Take a Snapshot	46
List Snapshots	46
Delete a Snapshot	46
Start the Server with a Snapshot	47
3.6.5. View Configuration Changes	47
Track and View Configuration Changes from the Management CLI	47
Track and View Configuration Changes from the Management Console	49
3.6.6. Property Replacement	49
Nested Expressions	50
Descriptor-Based Property Replacement	50
3.6.7. Using Git to Manage Configuration Data	51
Using a Local Git Repository	52
Using a Remote Git Repository	52
Publishing Remote Configuration Data When Using Git	54
Using Snapshots with Git	54
Taking Snapshots When Using Git	54
Listing Snapshots When Using Git	54
Deleting Snapshots When Using Git	54
3.7. FILE SYSTEM PATHS	55
3.7.1. View File System Paths	56
3.7.2. Override a Standard Path	56
Overriding a Managed Domain's Standard Paths	57
3.7.3. Add a Custom Path	57
3.8. DIRECTORY GROUPING	58
Directory Grouping by Server	58
Directory Grouping by Type	59
3.9. SYSTEM PROPERTIES	59
Passing a System Property to the Startup Script	60
Setting a System Property Using the Management CLI	60
Setting a System Property Using the Management Console	60
Setting a System Property Using JAVA_OPTS	60
3.10. MANAGEMENT AUDIT LOGGING	61

Standalone Server Audit Logging	61
Managed Domain Audit Logging	62
3.10.1. Enable Management Audit Logging	62
Enable Standalone Server Audit Logging	63
Enable Managed Domain Audit Logging	63
3.10.2. Enable JMX Management Audit Logging	63
Enable Standalone Server JMX Audit Logging	63
Enable Managed Domain JMX Audit Logging	63
3.10.3. Send Management Audit Logging to a Syslog Server	64
3.10.4. Read Audit Log Entries	65
3.11. SERVER LIFECYCLE EVENT NOTIFICATIONS	66
3.11.1. Monitor Server Lifecycle Events Using the Core Management Subsystem	66
3.11.2. Monitor Server Lifecycle Events Using JMX Notifications	69
CHAPTER 4. NETWORK AND PORT CONFIGURATION	72
4.1. INTERFACES	72
4.1.1. Default Interface Configurations	72
4.1.2. Configuring Interfaces	72
Add an Interface with a NIC Value	73
Add an Interface with Several Conditional Values	73
Update an Interface Attribute	73
Add an Interface to a Server in a Managed Domain	73
4.2. SOCKET BINDINGS	74
4.2.1. Management Ports	74
4.2.2. Default Socket Bindings	75
Standalone Server	75
Managed Domain	75
4.2.3. Configuring Socket Bindings	77
4.2.4. Viewing Socket Bindings and Open Ports for a Server	77
4.2.5. Port Offsets	78
4.3. IPV6 ADDRESSES	78
Configure the JVM Stack for IPv6 Addresses	78
Update Interface Declarations for IPv6 Addresses	79
CHAPTER 5. JBOSS EAP SECURITY	80
CHAPTER 6. JBOSS EAP CLASS LOADING	81
6.1. MODULES	81
Static Modules	81
Dynamic Modules	82
Predefined Modules	82
6.2. MODULE DEPENDENCIES	82
Optional Dependencies	83
Export a Dependency	83
Global Modules	83
6.3. CREATE A CUSTOM MODULE	83
Create a Custom Module Manually	83
Create a Custom Module Using the Management CLI	84
Add the Module as a Dependency	85
6.4. REMOVE A CUSTOM MODULE	85
Remove a Custom Module Manually	85
Remove a Custom Module Using the Management CLI	86
6.5. DEFINE GLOBAL MODULES	86
6.6. CONFIGURE SUBDEPLOYMENT ISOLATION	87

Enable Subdeployment Module Isolation for All Deployments	87
6.7. DEFINE AN EXTERNAL JBOSS EAP MODULE DIRECTORY	88
6.8. DYNAMIC MODULE NAMING CONVENTIONS	88
CHAPTER 7. DEPLOYING APPLICATIONS	89
7.1. DEPLOYING APPLICATIONS USING THE MANAGEMENT CLI	89
7.1.1. Deploy an Application to a Standalone Server Using the Management CLI	89
Deploy an Application	89
Undeploy an Application	90
Disable an Application	90
Enable an Application	90
List Deployments	90
7.1.2. Deploy an Application in a Managed Domain Using the Management CLI	91
Deploy an Application	91
Undeploy an Application	91
Disable an Application	92
Enable an Application	92
List Deployments	92
7.2. DEPLOYING APPLICATIONS USING THE MANAGEMENT CONSOLE	93
7.2.1. Deploy an Application to a Standalone Server Using the Management Console	93
Deploy an Application	93
Undeploy an Application	93
Disable an Application	93
Replace an Application	93
7.2.2. Deploy an Application in a Managed Domain Using the Management Console	93
Add an Application	94
Deploy an Application to a Server Group	94
Undeploy an Application from a Server Group	94
Remove an Application	94
Disable an Application	94
Replace an Application	94
7.3. DEPLOYING APPLICATIONS USING THE DEPLOYMENT SCANNER	95
7.3.1. Deploy an Application to a Standalone Server Using the Deployment Scanner	95
Deploy an Application	95
Undeploy an Application	95
Redeploy an Application	96
7.3.2. Configure the Deployment Scanner	96
Disable the Deployment Scanner	96
Change the Scan Interval	96
Change the Deployment Folder	96
Enable the Automatic Deployment of Exploded Content	96
Disable the Automatic Deployment of Zipped Content	96
Disable the Automatic Deployment of XML Content	97
7.3.3. Define a Custom Deployment Scanner	97
7.4. DEPLOYING APPLICATIONS USING MAVEN	97
7.4.1. Deploy an Application to a Standalone Server Using Maven	97
Deploy an Application	97
Undeploy an Application	98
7.4.2. Deploy an Application in a Managed Domain Using Maven	98
Deploy an Application	98
Undeploy an Application	99
7.5. DEPLOYING APPLICATIONS USING THE HTTP API	100
7.5.1. Deploy an Application to a Standalone Server Using the HTTP API	100

Deploy an Application	100
Undeploy an Application	100
7.5.2. Deploy an Application in a Managed Domain Using the HTTP API	100
Deploy an Application	100
Undeploy an Application	101
7.6. CUSTOMIZING DEPLOYMENT BEHAVIOR	101
7.6.1. Define a Custom Directory for Deployment Content	101
Define a Custom Directory for a Standalone Server	101
Define a Custom Directory for a Managed Domain	101
7.6.2. Control the Order of Deployments	102
7.6.3. Override Deployment Content	102
7.6.4. Using Rollout Plans	104
About Rollout Plans	104
Rollout Plan Syntax	105
Deploy Using a Rollout Plan	105
Deploy Using a Stored Rollout Plan	106
Remove a Stored Rollout Plan	106
Default Rollout Plan	106
7.7. MANAGING EXPLODED DEPLOYMENTS	107
Create an Empty Exploded Deployment	107
Explode an Existing Archive Deployment	107
Add Content to an Exploded Deployment	107
Remove Content from an Exploded Deployment	108
7.8. VIEWING DEPLOYMENT CONTENT	108
7.8.1. Browse Files in a Deployment	108
7.8.2. Read Deployment Content	109
7.8.2.1. Display the Contents of a File	109
7.8.2.2. Save the Contents of a File	110
CHAPTER 8. DOMAIN MANAGEMENT	111
8.1. ABOUT MANAGED DOMAINS	111
8.1.1. About the Domain Controller	112
8.1.2. About Host Controllers	112
8.1.3. About Process Controllers	113
8.1.4. About Server Groups	113
8.1.5. About Servers	113
8.2. NAVIGATING DOMAIN CONFIGURATIONS	113
Management Console	113
Management CLI	114
8.3. LAUNCHING A MANAGED DOMAIN	115
8.3.1. Start a Managed Domain	115
Start the Domain Controller	115
Start a Host Controller	116
8.3.2. Domain Controller Configuration	116
8.3.3. Host Controller Configuration	116
8.3.4. Configure the Name of a Host	117
8.4. MANAGING SERVERS	118
8.4.1. Configure Server Groups	118
Add a Server Group	118
Update a Server Group	119
Remove a Server Group	119
Server Group Attributes	119
8.4.2. Configure Servers	119

Add a Server	120
Update a Server	120
Remove a Server	120
Server Attributes	120
8.4.3. Start and Stop Servers	121
Start Servers	121
Stop Servers	121
Reload Servers	121
Kill Servers	121
8.5. DOMAIN CONTROLLER DISCOVERY AND FAILOVER	121
8.5.1. Configure Domain Discovery Options	122
8.5.2. Start a Host Controller with a Cached Domain Configuration	122
8.5.3. Promote a Host Controller to Act as Domain Controller	123
Cache the Domain Configuration	123
Promote a Host Controller to Be the Domain Controller	123
8.6. MANAGED DOMAIN SETUPS	124
8.6.1. Set Up a Managed Domain on a Single Machine	124
8.6.2. Set Up a Managed Domain on Two Machines	125
Create a Managed Domain on Two Machines	125
8.7. MANAGING MULTIPLE JBOSS EAP VERSIONS	126
8.7.1. Configure a JBoss EAP 7.x Domain Controller to Administer JBoss EAP 6 Instances	126
8.7.1.1. Add the JBoss EAP 6 Configuration to the JBoss EAP 7 Domain Controller	127
8.7.1.2. Update the Behavior for the JBoss EAP 6 Profiles	128
8.7.1.3. Set the Server Group for the JBoss EAP 6 Servers	129
8.7.1.4. Prevent the JBoss EAP 6 Instances from Receiving JBoss EAP 7 Updates	129
8.7.2. Configure a JBoss EAP 7.3 Domain Controller to Administer Previous Minor Releases of JBoss EAP	130
8.7.2.1. Add the JBoss EAP 7.2 Configuration to the JBoss EAP 7.3 Domain Controller	130
8.7.2.2. Set the Server Group for the JBoss EAP 7.2 Servers	132
8.7.2.3. Prevent the JBoss EAP 7.2 Instances from Receiving JBoss EAP 7.3 Updates	132
8.8. MANAGING JBOSS EAP PROFILES	133
8.8.1. About Profiles	133
8.8.2. Cloning a Profile	133
8.8.3. Creating Hierarchical Profiles	134
CHAPTER 9. CONFIGURING JVM SETTINGS	135
9.1. CONFIGURING JVM SETTINGS FOR A STANDALONE SERVER	135
9.2. CONFIGURING JVM SETTINGS FOR A MANAGED DOMAIN	135
9.2.1. Defining JVM Settings on a Host Controller	136
9.2.2. Applying JVM Settings to a Server Group	136
9.2.3. Applying JVM Settings to an Individual Server	136
9.3. DISPLAYING THE JVM STATUS	137
9.4. TUNING THE JVM	137
CHAPTER 10. MAIL SUBSYSTEM	138
10.1. CONFIGURING THE MAIL SUBSYSTEM	138
Configuring SMTP server for use in an application	138
10.2. CONFIGURING CUSTOM TRANSPORTS	138
10.3. USE A CREDENTIAL STORE FOR PASSWORDS	140
CHAPTER 11. LOGGING WITH JBOSS EAP	141
11.1. ABOUT SERVER LOGGING	141
11.1.1. Server Logging	141
11.1.2. Bootup Logging	141
11.1.2.1. View Bootup Errors	141

Examine the Server Log File	142
Read the Boot Errors from the Management CLI	142
11.1.3. Garbage Collection Logging	143
11.1.4. Default Log File Locations	143
11.1.5. Set the Default Locale of the Server	144
Set the Language	144
Set the Language and Country	144
Set the Server Locale Using the org.jboss.logging.locale Property	144
11.2. VIEWING LOG FILES	145
View Logs from the Management Console	145
View Logs from the Management CLI	145
11.3. ABOUT THE LOGGING SUBSYSTEM	146
11.3.1. Root Logger	146
11.3.2. Log Categories	147
11.3.3. Log Handlers	147
Log Handler Types	147
11.3.4. Log Levels	148
Supported Log Levels	148
11.3.5. Log Formatters	149
Pattern Formatter	150
JSON Formatter	150
XML Formatter	150
Custom Formatter	150
11.3.6. Filter Expressions	150
11.3.7. Implicit Logging Dependencies	152
11.4. CONFIGURING LOG CATEGORIES	152
Add a Log Category	152
Configure Log Category Settings	153
Assign a Handler	153
Remove a Log Category	153
11.5. CONFIGURING LOG HANDLERS	153
11.5.1. Configure a Console Log Handler	154
Add a Console Log Handler	154
Configure Console Log Handler Settings	154
Assign the Console Log Handler to a Logger	155
Remove a Console Log Handler	155
11.5.2. Configure a File Log Handler	156
Add a File Log Handler	156
Configure File Log Handler Settings	156
Assign the File Log Handler to a Logger	157
Remove a File Log Handler	157
11.5.3. Configure a Periodic Rotating Log Handler	157
Add a Periodic Log Handler	158
Configure Periodic Log Handler Settings	158
Assign the Periodic Log Handler to a Logger	159
Remove a Periodic Log Handler	159
11.5.4. Configure a Size Rotating Log Handler	160
Add a Size Log Handler	160
Configure Size Log Handler Settings	160
Assign the Size Log Handler to a Logger	162
Remove a Size Log Handler	162
11.5.5. Configure a Periodic Size Rotating Log Handler	162
Add a Periodic Size Log Handler	162

Configure Periodic Size Log Handler Settings	163
Assign the Periodic Size Log Handler to a Logger	164
Remove a Periodic Size Log Handler	164
11.5.6. Configure a Syslog Handler	165
Add a Syslog Handler	165
Configure Syslog Handler Settings	165
Assign the Syslog Handler to a Logger	166
Remove a Syslog Handler	166
11.5.7. Configure a Socket Log Handler	166
Add the Socket Binding	167
Add the Log Formatter	167
Add the Socket Log Handler	167
Configure Socket Log Handler Settings	167
Assign the Socket Log Handler to a Logger	168
Remove a Socket Log Handler	168
Send Log Messages Over a Socket Using SSL/TLS	168
11.5.8. Configure a Custom Log Handler	169
Add a Custom Log Handler	170
Configure Custom Log Handler Settings	170
Assign the Custom Log Handler to a Logger	171
Remove a Custom Log Handler	171
11.5.9. Configure an Async Log Handler	171
Add an Async Log Handler	171
Add a Sub-handler	172
Configure Async Log Handler Settings	172
Assign the Async Log Handler to a Logger	172
Remove an Async Log Handler	173
11.6. CONFIGURING THE ROOT LOGGER	173
Configure the Root Logger	173
11.7. CONFIGURING LOG FORMATTERS	173
11.7.1. Configure a Pattern Formatter	174
Create a Pattern Formatter	174
11.7.2. Configure a JSON Log Formatter	174
Add a JSON Log Formatter	175
Add a Logstash JSON Log Formatter	175
11.7.3. Configure an XML Log Formatter	176
Add an XML Log Formatter	176
Add a Key Override XML Log Formatter	176
11.7.4. Configure a Custom Log Formatter	177
Configure a Custom Log Formatter	177
Example Custom XML Formatter	178
Configure a Custom Log Formatter Using the Management Console	178
11.8. ABOUT APPLICATION LOGGING	179
11.8.1. Per-deployment Logging	179
11.8.1.1. Disable Per-deployment Logging	179
11.8.2. Logging Profiles	179
11.8.2.1. Configure a Logging Profile	180
Creating and Configuring a Logging Profile	180
11.8.2.2. Example Logging Profile Configuration	181
11.8.3. Viewing Deployment Logging Configuration	182
11.9. TUNING THE LOGGING SUBSYSTEM	183

CHAPTER 12. DATASOURCE MANAGEMENT	184
--	------------

12.1. ABOUT JBOSS EAP DATASOURCES	184
About JDBC	184
Supported Databases	184
Datasource Types	184
The ExampleDS datasource	184
12.2. JDBC DRIVERS	184
12.2.1. Installing a JDBC Driver as a Core Module	184
12.2.1.1. Add the JDBC Driver as a Core Module	185
12.2.1.2. Register the JDBC Driver	185
12.2.2. Installing a JDBC Driver as a JAR Deployment	186
Update a JDBC Driver JAR to be JDBC 4-Compliant	187
12.2.3. JDBC Driver Download Locations	187
12.2.4. Access Vendor-Specific Classes	188
Using the MANIFEST.MF File	188
Using a jboss-deployment-structure.xml File	189
12.3. CREATING DATASOURCES	189
12.3.1. Create a Non-XA Datasource	189
Defining a Non-XA Datasource Using the Management Console	189
Defining a Non-XA Datasource Using the Management CLI	190
Datasource Parameters	190
12.3.2. Create an XA Datasource	191
Defining an XA Datasource Using the Management Console	191
Defining an XA Datasource Using the Management CLI	191
Datasource Parameters	192
12.4. MODIFYING DATASOURCES	192
12.4.1. Modify a Non-XA Datasource	192
12.4.2. Modify an XA Datasource	193
12.5. REMOVING DATASOURCES	193
12.5.1. Remove a Non-XA Datasource	193
12.5.2. Remove an XA Datasource	194
12.6. TESTING DATASOURCE CONNECTIONS	194
Test a Datasource Connection Using the Management CLI	194
Test a Datasource Connection Using the Management Console	194
12.7. FLUSHING DATASOURCE CONNECTIONS	194
12.8. XA DATASOURCE RECOVERY	195
12.8.1. Configuring XA Recovery	195
12.8.2. Vendor-Specific XA Recovery	196
Vendor-Specific Configuration	196
Known Issues	197
12.9. DATABASE CONNECTION VALIDATION	199
12.10. DATASOURCE SECURITY	201
Secure a Datasource Using a Security Domain	201
Secure a Datasource Using a Password Vault	202
Secure a Datasource Using a Credential Store	202
Secure a Datasource Using an Authentication Context	202
Secure a Datasource using Kerberos	203
12.11. DATASOURCE STATISTICS	205
12.11.1. Enabling Datasource Statistics	205
Enable Datasource Statistics Using the Management CLI	206
Enable Datasource Statistics Using the Management Console	206
12.11.2. Viewing Datasource Statistics	206
View Datasource Statistics Using the Management CLI	206
View Datasource Statistics Using the Management Console	207

12.12. DATASOURCE TUNING	207
12.13. CAPACITY POLICIES	207
MaxPoolSize Incrementer Policy	208
Size Incrementer Policy	208
Watermark Incrementer Policy	208
MinPoolSize Decrementer Policy	209
Size Decrementer Policy	209
TimedOut Decrementer Policy	209
TimedOut/FIFO Decrementer Policy	209
Watermark Decrementer Policy	209
12.14. ENLISTMENT TRACING	210
12.15. EXAMPLE DATASOURCE CONFIGURATIONS	210
12.15.1. Example MySQL Datasource	210
Example: MySQL Datasource Configuration	210
Example: MySQL JDBC Driver module.xml File	211
Example Management CLI Commands	211
12.15.2. Example MySQL XA Datasource	212
Example: MySQL XA Datasource Configuration	212
Example: MySQL JDBC Driver module.xml File	212
Example Management CLI Commands	213
12.15.3. Example PostgreSQL Datasource	213
Example: PostgreSQL Datasource Configuration	214
Example: PostgreSQL JDBC Driver module.xml File	214
Example Management CLI Commands	214
12.15.4. Example PostgreSQL XA Datasource	215
Example: PostgreSQL XA Datasource Configuration	215
Example: PostgreSQL JDBC Driver module.xml File	216
Example Management CLI Commands	216
12.15.5. Example Oracle Datasource	217
Example: Oracle Datasource Configuration	217
Example: Oracle JDBC Driver module.xml File	218
Example Management CLI Commands	218
12.15.6. Example Oracle XA Datasource	219
Example: Oracle XA Datasource Configuration	219
Example: Oracle JDBC Driver module.xml File	220
Example Management CLI Commands	220
12.15.7. Example Microsoft SQL Server Datasource	221
Example: Microsoft SQL Server Datasource Configuration	221
Example: Microsoft SQL Server JDBC Driver module.xml File	221
Example Management CLI Commands	221
12.15.8. Example Microsoft SQL Server XA Datasource	222
Example: Microsoft SQL Server XA Datasource Configuration	222
Example: Microsoft SQL Server JDBC Driver module.xml File	223
Example Management CLI Commands	223
12.15.9. Example IBM DB2 Datasource	224
Example: IBM DB2 Datasource Configuration	224
Example: IBM DB2 JDBC Driver module.xml File	225
Example Management CLI Commands	225
12.15.10. Example IBM DB2 XA Datasource	226
Example: IBM DB2 XA Datasource Configuration	226
Example: IBM DB2 JDBC Driver module.xml File	227
Example Management CLI Commands	227
12.15.11. Example Sybase Datasource	228

Example: Sybase Datasource Configuration	228
Example: Sybase JDBC Driver module.xml File	228
Example Management CLI Commands	229
12.15.12. Example Sybase XA Datasource	229
Example: Sybase XA Datasource Configuration	229
Example: Sybase JDBC Driver module.xml File	230
Example Management CLI Commands	230
12.15.13. Example MariaDB Datasource	231
Example: MariaDB Datasource Configuration	231
Example: MariaDB JDBC Driver module.xml File	232
Example Management CLI Commands	232
12.15.14. Example MariaDB XA Datasource	233
Example: MariaDB XA Datasource Configuration	233
Example: MariaDB JDBC Driver module.xml File	233
Example Management CLI Commands	234
12.15.15. Example MariaDB Galera Cluster Datasource	234
Example: MariaDB Galera Cluster Datasource Configuration	235
Example: MariaDB JDBC Driver module.xml File	235
Example Management CLI Commands	235
CHAPTER 13. DATASOURCE MANAGEMENT WITH AGROAL	237
13.1. ABOUT THE AGROAL DATASOURCES SUBSYSTEM	237
13.2. ENABLING THE AGROAL DATASOURCES SUBSYSTEM	237
13.3. INSTALLING A JDBC DRIVER AS A CORE MODULE FOR AGROAL DATASOURCES	238
13.3.1. Add the JDBC Driver as a Core Module	238
13.3.2. Register the JDBC Driver for Agroal Datasources	239
13.4. CONFIGURING AGROAL NON-XA DATASOURCES	239
13.4.1. Create an Agroal Datasource	239
13.4.2. Remove an Agroal Datasource	239
13.5. CONFIGURING AGROAL XA DATASOURCES	239
13.5.1. Create an Agroal XA Datasource	240
13.5.2. Remove an Agroal XA Datasource	240
13.6. EXAMPLE AGROAL DATASOURCE CONFIGURATIONS	240
13.6.1. Example MySQL Agroal Datasource	240
Example: MySQL Agroal Datasource Configuration	240
Example: MySQL JDBC Driver module.xml File	241
Example Management CLI Commands	241
13.6.2. Example MySQL Agroal XA Datasource	241
Example: MySQL Agroal XA Datasource Configuration	242
Example: MySQL JDBC Driver module.xml File	242
Example Management CLI Commands	242
CHAPTER 14. CONFIGURING THE TRANSACTIONS SUBSYSTEM	244
CHAPTER 15. ORB CONFIGURATION	245
15.1. ABOUT COMMON OBJECT REQUEST BROKER ARCHITECTURE (CORBA)	245
15.2. CONFIGURING THE ORB FOR JTS	245
Configure the ORB Using the Management CLI	245
Enable the Security Interceptors	245
Enable Transactions in the IIOP Subsystem	245
Enable JTS in the Transactions Subsystem	246
Configure the ORB Using the Management Console	246
15.3. CONFIGURE IIOP TO USE SSL/TLS WITH THE ELYTRON SUBSYSTEM	246

CHAPTER 16. JAKARTA CONNECTORS MANAGEMENT	248
16.1. ABOUT JAKARTA CONNECTORS	248
16.2. ABOUT RESOURCE ADAPTERS	248
16.3. CONFIGURING THE JCA SUBSYSTEM	248
Archive Validation	249
Bean Validation	250
Work Managers	250
Distributed Work Managers	251
Bootstrap Contexts	253
Cached Connection Manager	253
16.4. CONFIGURING RESOURCE ADAPTERS	253
16.4.1. Deploy a Resource Adapter	253
Deploy a Resource Adapter using the Management CLI	253
Deploy a Resource Adapter using the Management Console	254
Deploy a Resource Adapter Using the Deployment Scanner	254
16.4.2. Configure a Resource Adapter	254
Add the Resource Adapter Configuration	254
Configure the Resource Adapter Settings	254
Activate the Resource Adapter	255
16.4.3. Configure Resource Adapters to Use the Elytron Subsystem	256
Container-managed Sign-On	256
Security Inflow	256
16.4.4. Deploy and Configure the IBM MQ Resource Adapter	259
16.4.5. Deploy and Configure the Generic JMS Resource Adapter	260
16.5. CONFIGURE MANAGED CONNECTION POOLS	260
16.6. VIEW CONNECTION STATISTICS	260
16.7. FLUSHING RESOURCE ADAPTER CONNECTIONS	261
16.8. TUNING THE RESOURCE ADAPTERS SUBSYSTEM	261
CHAPTER 17. CONFIGURING THE WEB SERVER (UNDERTOW)	262
17.1. UNDERTOW SUBSYSTEM OVERVIEW	262
Default Undertow Subsystem Configuration	262
Using Elytron with Undertow Subsystem	263
Runtime Information	264
17.2. CONFIGURING BUFFER CACHES	265
Default Undertow Subsystem Configuration	265
Updating an Existing Buffer Cache	265
Creating a New Buffer Cache	265
Deleting a Buffer Cache	265
17.3. CONFIGURING BYTE BUFFER POOLS	265
Updating an Existing Byte Buffer Pool	266
Creating a Byte Buffer Pool	266
Deleting a Byte Buffer Pool	266
17.4. CONFIGURING A SERVER	266
Default Undertow Subsystem Configuration	267
Updating an Existing Server	267
Creating a New Server	267
Deleting a Server	267
17.4.1. Access Logging	268
17.4.1.1. Standard Access Logging	268
17.4.1.2. Console Access Logging	270
17.5. CONFIGURING A SERVLET CONTAINER	277
Default Undertow Subsystem Configuration	277

Updating an Existing Servlet Container	277
Creating a New Servlet Container	277
Deleting a Servlet Container	278
17.6. CONFIGURING A SERVLET EXTENSION	278
17.7. CONFIGURING HANDLERS	278
Default Undertow Subsystem Configuration	278
Using WebDAV for Static Resources	279
Updating an Existing File Handler	279
Creating a New File Handler	279
Deleting a File Handler	279
17.8. CONFIGURING FILTERS	280
Updating an Existing Filter	280
Creating a New Filter	280
Deleting a Filter	280
17.8.1. Configuring the buffer-request Handler	281
17.9. CONFIGURE THE DEFAULT WELCOME WEB APPLICATION	281
Default Undertow Subsystem Configuration	282
Change the welcome-content File Handler	282
Change the default-web-module	283
Disable the Default Welcome Web Application	283
17.10. CONFIGURING HTTPS	283
17.11. CONFIGURING HTTP SESSION TIMEOUT	283
Configuring the Default Session Timeout	283
17.12. CONFIGURING HTTP-ONLY SESSION MANAGEMENT COOKIES	284
Configuring host-only for the Servlet Container Session Cookie	284
Configuring host-only for the Host Single Sign-On	284
17.13. CONFIGURING HTTP/2	284
ALPN support when using HTTP/2	285
Verify HTTP/2 is Being Used	286
17.14. CONFIGURING A REQUESTDUMPING HANDLER	286
17.14.1. Configuring a RequestDumping Handler on the Server	286
Create a new Expression Filter with the RequestDumping Handler	286
Enable the Expression Filter in the Undertow Web Server	286
Configuring a RequestDumping Handler for Specific URLs	286
17.14.2. Configuring a RequestDumping Handler within an Application	287
17.15. CONFIGURING COOKIE SECURITY	287
17.16. TUNING THE UNDERTOW SUBSYSTEM	287
CHAPTER 18. CONFIGURING REMOTING	288
18.1. ABOUT THE REMOTING SUBSYSTEM	288
Default Remoting Subsystem Configuration	288
The Remoting Endpoint	288
Connector	288
Outbound Connections	288
Additional Configuration	288
18.2. CONFIGURING THE ENDPOINT	288
Updating the Existing Endpoint Configuration	289
Creating a New Endpoint Configuration	289
Deleting an Endpoint Configuration	289
18.3. CONFIGURING A CONNECTOR	289
Updating the Existing Connector Configuration	289
Creating a New Connector	289
Deleting a Connector	289

18.4. CONFIGURING AN HTTP CONNECTOR	290
Updating the Existing HTTP Connector Configuration	290
Creating a New HTTP Connector	290
Deleting an HTTP Connector	290
18.5. CONFIGURING AN OUTBOUND CONNECTION	290
Updating an Existing Outbound Connection	290
Creating a New Outbound Connection	290
Deleting an Outbound Connection	290
18.6. CONFIGURING A REMOTE OUTBOUND CONNECTION	291
18.7. CONFIGURING A LOCAL OUTBOUND CONNECTION	291
Updating an Existing Local Outbound Connection	291
Creating a New Local Outbound Connection	291
Deleting a Local Outbound Connection	291
18.8. ADDITIONAL REMOTING CONFIGURATION	291
CHAPTER 19. CONFIGURING THE IO SUBSYSTEM	294
19.1. IO SUBSYSTEM OVERVIEW	294
Default IO Subsystem Configuration	294
19.2. CONFIGURING A WORKER	294
Updating an Existing Worker	294
Creating a New Worker	294
Deleting a Worker	294
19.3. CONFIGURING A BUFFER POOL	294
Updating an Existing Buffer Pool	295
Creating a Buffer Pool	295
Deleting a Buffer Pool	295
19.4. TUNING THE IO SUBSYSTEM	295
CHAPTER 20. CONFIGURING WEB SERVICES	296
CHAPTER 21. JAKARTA SERVER FACES CONFIGURATION	297
21.1. MULTIPLE JAKARTA SERVER FACES IMPLEMENTATION OF JAKARTA SERVER FACES	297
21.1.1. Installing a Jakarta Server Faces Implementation	297
Add the Jakarta Server Faces Implementation JAR File	297
Add the Jakarta Server Faces API JAR File	297
Add the Jakarta Server Faces Injection JAR File	298
Add the commons-digester JAR File for MyFaces	298
Set the Default Jakarta Server Faces Implementation	298
21.1.2. Multi-Jakarta Server Faces Implementation Support	298
21.1.2.1. Working of the Multi-Jakarta Server Faces Implementation	299
21.1.2.2. Changing the Default Jakarta Server Faces Implementation	299
21.2. DISALLOWING DOCTYPE DECLARATIONS	299
CHAPTER 22. CONFIGURING BATCH APPLICATIONS	301
22.1. CONFIGURING BATCH JOBS	301
22.1.1. Configure Batch Job Repositories	301
Add an In-memory Job Repository	301
Add a JDBC Job Repository	301
Set a Default Job Repository	302
22.1.2. Configure Batch Thread Pools	302
Configure a Thread Pool	302
Use a Thread Factory	302
Set a Default Thread Pool	303
View Thread Pool Statistics	303

22.2. MANAGING BATCH JOBS	303
Manage Batch Jobs from the Management CLI	303
Restart a Batch Job	303
Start a Batch Job	304
Stop a Batch Job	304
View Batch Job Execution Details	304
Manage Batch Jobs from the Management Console	305
Restart a Batch Job	305
Start a Batch Job	305
Stop a Batch Job	305
View Batch Job Execution Details	305
22.3. CONFIGURE SECURITY FOR BATCH JOBS	305
CHAPTER 23. CONFIGURING THE NAMING SUBSYSTEM	306
23.1. ABOUT THE NAMING SUBSYSTEM	306
23.2. CONFIGURING GLOBAL BINDINGS	306
Configuring Simple Bindings	306
Binding Object Factories	307
Binding External Contexts	308
Binding Lookup Aliases	309
23.3. DYNAMICALLY CHANGE JNDI BINDINGS	309
23.4. CONFIGURING THE REMOTE JNDI INTERFACE	310
CHAPTER 24. CONFIGURING HIGH AVAILABILITY	311
24.1. INTRODUCTION TO HIGH AVAILABILITY	311
24.2. CLUSTER COMMUNICATION WITH JGROUPS	312
24.2.1. About JGroups	312
24.2.2. Switch the Default JGroups Channel to Use TCP	312
24.2.3. Configure TCPPING	313
24.2.4. Configure TCPGOSSIP	314
24.2.5. Configure JDBC_PING	316
Related information	316
24.2.6. Binding JGroups to a Network Interface	317
24.2.7. Securing a Cluster	317
24.2.7.1. Configuring Authentication	317
AUTH with a Simple Token	317
AUTH with a Digest Algorithm Token	317
AUTH with an X509 Token	318
24.2.7.2. Configuring Encryption	318
Using Symmetric Encryption	318
Using Symmetric Encryption by Directly Referencing the Keystore	319
Using Symmetric Encryption with Elytron	319
Using Asymmetric Encryption	320
Using Asymmetric Encryption by Generating a Secret Key	320
Using Asymmetric Encryption with Elytron	321
24.2.8. Configure JGroups Thread Pools	322
24.2.9. Configure JGroups Send and Receive Buffers	323
Resolving Buffer Size Warnings	323
Configuring JGroups Buffer Sizes	323
24.2.10. Tuning the JGroups Subsystem	324
24.2.11. JGroups Troubleshooting	324
24.2.11.1. Nodes Do Not Form a Cluster	324
24.2.11.2. Causes of Missing Heartbeats in Failure Detection	325

24.3. INFINISPAN	325
24.3.1. About Infinispan	325
24.3.2. Cache Containers	326
24.3.2.1. Configure Cache Containers	327
Configure Caches Using the Management Console	327
Configure Caches Using the Management CLI	327
Change the Default EJB Cache Container	328
Eviction Feature in Hibernate Cache Container	329
24.3.3. Clustering Modes	329
Cache Modes	329
Synchronous and Asynchronous Replication	330
24.3.3.1. Configure the Cache Mode	330
Change to Replication Cache Mode	331
Change to Distribution Cache Mode	331
Change to Scattered Cache Mode	332
24.3.3.2. Cache Strategy Performance	332
24.3.4. State Transfer	333
24.3.5. Configure Infinispan Thread Pools	333
24.3.6. Infinispan Statistics	334
24.3.6.1. Enable Infinispan Statistics	334
24.3.7. Infinispan Partition Handling	335
24.3.7.1. Split Brain	336
24.3.7.2. Configuring Partition Handling	336
24.3.8. Configuring Remote Cache Containers	336
24.3.8.1. Creating a Remote Cache Container	337
24.3.8.2. Enabling Statistics for a Remote Cache Container	337
24.3.9. Externalize HTTP Sessions to Red Hat Data Grid	338
Securing a Remote Cache Container	340
24.3.10. Externalize HTTP Sessions to Red Hat Data Grid Using a Remote Store	340
24.4. CONFIGURING JBOSS EAP AS A FRONT-END LOAD BALANCER	342
24.4.1. Configure Undertow as a Load Balancer Using mod_cluster	343
Configure the mod_cluster Front-end Load Balancer	343
Multiple mod_cluster Configurations	344
24.4.2. Enabling Ranked Session Affinity in Your Load Balancer	344
24.4.3. Configure Undertow as a Static Load Balancer	344
24.5. USING AN EXTERNAL WEB SERVER AS A PROXY SERVER	345
24.5.1. Overview of HTTP Connectors	346
24.5.2. Apache HTTP Server	347
24.5.2.1. Installing Apache HTTP Server	347
24.5.3. Accepting Requests from External Web Servers	347
Update JBoss EAP Configuration	348
24.6. THE MOD_CLUSTER HTTP CONNECTOR	348
24.6.1. Configure mod_cluster in Apache HTTP Server	349
Configure mod_cluster	349
24.6.2. Disable Advertising for mod_cluster	350
24.6.3. Configure a mod_cluster Worker Node	351
Configure a Worker Node	351
24.6.4. Configure the mod_cluster fail_on_status Parameter	356
24.6.5. Migrate Traffic Between Clusters	356
Upgrade Process for Clusters - Load-Balancing Groups	356
Default Load-Balancing Group	358
Using the Management CLI	358
24.7. APACHE MOD_JK HTTP CONNECTOR	358

24.7.1. Configure mod_jk in Apache HTTP Server	359
24.7.2. Configure JBoss EAP to Communicate with mod_jk	362
24.8. APACHE MOD_PROXY HTTP CONNECTOR	362
24.8.1. Configure mod_proxy in Apache HTTP Server	362
Add a Non-load-balancing Proxy	363
Add a Load-balancing Proxy	363
Enable Sticky Sessions	364
24.8.2. Configure JBoss EAP to Communicate with mod_proxy	364
24.9. MICROSOFT ISAPI CONNECTOR	364
24.9.1. Configure Microsoft IIS to Use the ISAPI Connector	365
24.9.2. Configure the ISAPI Connector to Send Client Requests to JBoss EAP	366
Create Property Files and Set Up Redirection	366
24.9.3. Configure the ISAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers	368
Balance Client Requests Across Multiple Servers	368
24.10. ORACLE NSAPI CONNECTOR	370
24.10.1. Configure Oracle iPlanet Web Server to use the NSAPI Connector	371
24.10.2. Configure the NSAPI Connector to Send Client Requests to JBoss EAP	372
Set Up the Basic HTTP Connector	372
24.10.3. Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers	374
Configure the Connector for Load Balancing	374
CHAPTER 25. ECLIPSE MICROPROFILE	376
25.1. USING ECLIPSE MICROPROFILE CONFIG TO MANAGE CONFIGURATION	376
25.1.1. About the MicroProfile Config SmallRye Subsystem	376
25.1.2. ConfigSources Supported by the MicroProfile Config SmallRye Subsystem	376
Obtaining ConfigSource Configuraton from Properties	377
Obtaining ConfigSource Configuraton from a Directory	377
Obtaining ConfigSource Configuraton from a ConfigSource Class	378
Obtaining ConfigSource Configuraton from a ConfigSourceProvider Class	378
25.1.3. Deploying Applications that Access ConfigSources	379
25.2. TRACING REQUESTS WITH THE MICROPROFILE OPENTRACING SMALLRYE SUBSYSTEM	379
25.2.1. About Eclipse MicroProfile OpenTracing	379
25.2.2. About the MicroProfile OpenTracing SmallRye Subsystem	379
25.2.3. Configuring the MicroProfile OpenTracing SmallRye Subsystem	380
25.3. MONITOR SERVER HEALTH USING ECLIPSE MICROPROFILE HEALTH	381
25.3.1. MicroProfile Health SmallRye Subsystem	381
25.3.2. Health Check Using the Management CLI Examples	381
25.3.3. Examining the Health Check Using the Management Console	382
25.3.4. Examine the Health Check Using the HTTP Endpoint	382
25.3.5. Global Status When Probes are not Defined	382
25.3.6. Enable Authentication for the HTTP Endpoint	383
25.4. ECLIPSE MICROPROFILE METRICS	383
25.4.1. About the MicroProfile Metrics Subsystem	384
25.4.2. Examine Metrics Using the HTTP Endpoint	384
25.4.3. Configure MicroProfile Metrics using the Management Console	385
25.4.4. Enable Authentication for the HTTP Endpoint	385
APPENDIX A. REFERENCE MATERIAL	387
A.1. SERVER RUNTIME ARGUMENTS	387
A.2. RPM SERVICE CONFIGURATION FILES	390
A.3. RPM SERVICE CONFIGURATION PROPERTIES	391
A.4. OVERVIEW OF JBOSS EAP SUBSYSTEMS	392
A.5. ADD-USER UTILITY ARGUMENTS	397

A.6. MANAGEMENT AUDIT LOGGING ATTRIBUTES	398
A.7. INTERFACE ATTRIBUTES	400
A.8. SOCKET BINDING ATTRIBUTES	402
A.9. DEFAULT SOCKET BINDINGS	404
A.10. MODULE COMMAND ARGUMENTS	407
A.11. DEPLOYMENT SCANNER MARKER FILES	409
A.12. DEPLOYMENT SCANNER ATTRIBUTES	410
A.13. MANAGED DOMAIN JVM CONFIGURATION ATTRIBUTES	411
A.14. MAIL SUBSYSTEM ATTRIBUTES	412
A.15. ROOT LOGGER ATTRIBUTES	415
A.16. LOG CATEGORY ATTRIBUTES	415
A.17. LOG HANDLER ATTRIBUTES	416
A.18. LOG FORMATTER ATTRIBUTES	423
A.19. DATASOURCE CONNECTION URLS	426
A.20. DATASOURCE ATTRIBUTES	426
A.21. DATASOURCE STATISTICS	435
A.22. AGROAL DATASOURCE ATTRIBUTES	439
A.23. TRANSACTION MANAGER CONFIGURATION OPTIONS	440
A.24. IIOP SUBSYSTEM ATTRIBUTES	444
A.25. RESOURCE ADAPTER ATTRIBUTES	446
A.26. RESOURCE ADAPTER STATISTICS	452
A.27. UNDERTOW SUBSYSTEM ATTRIBUTES	453
Application Security Domain Attributes	454
application-security-domain Attributes	454
single-sign-on Attributes	455
Buffer Cache Attributes	455
Byte Buffer Pool Attributes	456
Servlet Container Attributes	457
servlet-container Attributes	457
mime-mapping Attributes	459
crawler-session-management Attributes	459
jsp Attributes	459
persistent-sessions Attributes	460
session-cookie Attributes	461
websockets Attributes	461
welcome-file Attributes	462
Filter Attributes	462
custom-filter Filters	462
error-page Filters	462
expression-filter Filters	463
gzip Filters	463
mod-cluster Filters	463
request-limit Filters	467
response-header Filters	468
rewrite Filters	468
Handler Attributes	468
file Attributes	468
Using WebDAV for Static Resources	469
reverse-proxy attributes	469
Server Attributes	470
server Attributes	471
ajp-listener Attributes	471
host Attributes	474

filter-ref Attributes	474
location Attributes	474
filter-ref Attributes	475
access-log Attributes	475
console-access-log Attributes	476
http-invoker Attributes	477
single-sign-on Attributes	477
http-listener Attributes	478
https-listener Attributes	482
A.28. UNDERTOW SUBSYSTEM STATISTICS	486
A.29. DEFAULT BEHAVIOR OF HTTP METHODS	487
A.30. REMOTING SUBSYSTEM ATTRIBUTES	487
Connector Attributes	490
HTTP Connector Attributes	493
Outbound Connection Attributes	494
Remote Outbound Connection	494
Local Outbound Connection Attributes	495
A.31. IO SUBSYSTEM ATTRIBUTES	495
A.32. JAKARTA SERVER FACES MODULE TEMPLATES	496
Example: Mojarra Jakarta Server Faces Implementation JAR module.xml	497
Example: MyFaces Jakarta Server Faces Implementation JAR module.xml	497
Example: Mojarra Jakarta Server Faces API JAR module.xml	498
Example: MyFaces Jakarta Server Faces API JAR module.xml	499
Example: Mojarra Jakarta Server Faces Injection JAR module.xml	499
Example: MyFaces Jakarta Server Faces Injection JAR module.xml	500
Example: MyFaces commons-digester JAR module.xml	501
A.33. JGROUPS SUBSYSTEM ATTRIBUTES	501
Channel Attributes	502
channel Attributes	502
Stack Attributes	502
stack Attributes	503
protocol Attributes	503
relay Attributes	503
remote-site Attributes	503
transport Attributes	504
thread-pool Attributes	505
A.34. JGROUPS PROTOCOLS	505
Generic Protocol Attributes	507
Authentication Protocols	507
AUTH Attributes	507
Token Types	507
SASL Attributes	508
Discovery Protocols	509
AZURE_PING Attributes	509
JDBC_PING Attributes	509
S3_PING Attributes	510
TCPGOSSIP Attributes	510
TCPPING Attributes	511
Encrypt Protocols	511
ASYM_ENCRYPT Attributes	511
SYM_ENCRYPT Attributes	512
Failure Detection Protocols	512
Flow Control Protocols	512

Group Membership Protocols	512
Merge Protocols	512
Message Stability	513
Reliable Message Transmission	513
Deprecated Protocols	513
A.35. MICROPROFILE CONFIG SMALLRYE SUBSYSTEM ATTRIBUTES	513
A.36. APACHE HTTP SERVER MOD_CLUSTER DIRECTIVES	514
A.37. MODCLUSTER SUBSYSTEM ATTRIBUTES	517
A.38. MOD_JK WORKER PROPERTIES	522
A.39. SECURITY MANAGER SUBSYSTEM ATTRIBUTES	525
A.40. INSTALL OPENSSL FROM JBOSS CORE SERVICES	525
Using JBoss Core Services OpenSSL ZIP File Distribution	525
Using JBoss Core Services OpenSSL RPM Distribution	526
A.41. CONFIGURE JBOSS EAP TO USE OPENSSL	527
A.42. PLATFORM MODULES PROVIDED FOR JAVA 8	528

CHAPTER 1. INTRODUCTION

Before using this guide to configure JBoss EAP, it is assumed that the latest version of JBoss EAP has been downloaded and installed. For installation instructions, see the JBoss EAP [Installation Guide](#).



IMPORTANT

Since the installation location of JBoss EAP will vary between host machines, this guide refers to the installation location as ***EAP_HOME***. The actual location of the JBoss EAP installation should be used instead of ***EAP_HOME*** when performing administrative tasks.

CHAPTER 2. STARTING AND STOPPING JBOSS EAP

2.1. STARTING JBOSS EAP

JBoss EAP is supported on Red Hat Enterprise Linux, Windows Server, and Oracle Solaris, and runs in either a standalone server or managed domain operating mode. The specific command to start JBoss EAP depends on the underlying platform and the desired operating mode.

Servers are initially started in a suspended state and will not accept any requests until all required services have started, at which time the servers are placed into a normal running state and can start accepting requests.

Start JBoss EAP as a Standalone Server

```
$ EAP_HOME/bin/standalone.sh
```



NOTE

For Windows Server, use the ***EAP_HOME\bin\standalone.bat*** script.

This startup script uses the ***EAP_HOME/bin/standalone.conf*** file, or ***standalone.conf.bat*** for Windows Server, to set some default preferences, such as JVM options. You can customize the settings in this file.

JBoss EAP uses the ***standalone.xml*** configuration file by default, but can be started using a different one. For details on the available standalone configuration files and how to use them, see the [Standalone Server Configuration Files](#) section.

For a complete listing of all available startup script arguments and their purposes, use the **--help** argument or see the [Server Runtime Arguments](#) section.

Start JBoss EAP in a Managed Domain

The domain controller must be started before the servers in any of the server groups in the domain. Use this script to first start the domain controller, and then for each associated host controller.

```
$ EAP_HOME/bin/domain.sh
```



NOTE

For Windows Server, use the ***EAP_HOME\bin\domain.bat*** script.

This startup script uses the ***EAP_HOME/bin/domain.conf*** file, or ***domain.conf.bat*** for Windows Server, to set some default preferences, such as JVM options. You can customize the settings in this file.

JBoss EAP uses the ***host.xml*** host configuration file by default, but can be started using a different one. For details on the available managed domain configuration files and how to use them, see the [Managed Domain Configuration Files](#) section.

When setting up a managed domain, additional arguments will need to be passed into the startup script. For a complete listing of all available startup script arguments and their purposes, use the **--help** argument or see the [Server Runtime Arguments](#) section.

2.2. STOPPING JBOSS EAP

The way that you stop JBoss EAP depends on how it was started.

Stop an Interactive Instance of JBoss EAP

Press **Ctrl+C** in the terminal where JBoss EAP was started.

Stop a Background Instance of JBoss EAP

Use the management CLI to connect to the running instance and shut down the server.

1. Launch the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

2. Issue the **shutdown** command.

```
shutdown
```



NOTE

When running in a managed domain, you must specify the host name to shut down by using the **--host** argument with the **shutdown** command.

2.3. RUNNING JBOSS EAP IN ADMIN-ONLY MODE

JBoss EAP has the ability to be started in *admin-only* mode. This enables JBoss EAP to run and accept management requests but not start other runtime services or accept end user requests. Admin-only mode is available in both [standalone servers](#) as well as [managed domains](#).

Running a Standalone Server in Admin-only Mode

Start the Server in Admin-only Mode

To start a JBoss EAP instance in admin-only mode, use the **--start-mode=admin-only** runtime argument when starting the JBoss EAP instance.

```
$ EAP_HOME/bin/standalone.sh --start-mode=admin-only
```

Check If the Server is Running in Admin-only Mode

Use the following command to check the running mode of the server. The result will be **ADMIN_ONLY** if the server is running in admin-only mode.

```
:read-attribute(name=running-mode)
{
  "outcome" => "success",
  "result" => "ADMIN_ONLY"
}
```



NOTE

Additionally, you can check the initial running mode in which JBoss EAP was launched by using the following command.

```
/core-service=server-environment:read-attribute(name=initial-running-mode)
```

Reload in a Different Mode from the Management CLI

In addition to stopping and starting a JBoss EAP instance with a different runtime switch, the management CLI may also be used to reload it in a different mode.

To reload the server in admin-only mode:

```
reload --start-mode=admin-only
```

To reload the server in normal mode:

```
reload --start-mode=normal
```

Note that if the server was started in admin-only mode and no **--start-mode** argument is specified to the **reload** command, the server will be started in normal mode.

Running a Managed Domain in Admin-only Mode

In a managed domain, if a domain controller is started in admin-only mode, it will not accept incoming connections from slave host controllers.

Start a Host Controller in Admin-only Mode

Pass in the **--admin-only** runtime argument to start a host controller in admin-only mode.

```
$ EAP_HOME/bin/domain.sh --admin-only
```

Check If a Host Controller is Running in Admin-only Mode

Use the following command to check the running mode of a host controller. The result will be **ADMIN_ONLY** if the host controller is running in admin-only mode.

```
/host=HOST_NAME:read-attribute(name=running-mode)
{
  "outcome" => "success",
  "result" => "ADMIN_ONLY"
}
```

Reload in a Different Mode from the Management CLI

In addition to stopping and starting a host controller with a different runtime switch, the management CLI may also be used to reload it in a different mode.

To reload the host controller in admin-only mode:

```
reload --host=HOST_NAME --admin-only=true
```

To reload a host controller in normal mode:

```
reload --host=HOST_NAME --admin-only=false
```

Note that if the host controller was started in admin-only mode and no **--admin-only** argument is specified to the **reload** command, the host controller will be started in normal mode.

2.4. SUSPEND AND SHUT DOWN JBOSS EAP GRACEFULLY

JBoss EAP can be suspended or shut down gracefully. This allows active requests to complete normally, without accepting any new requests. A timeout value specifies how long that the suspend or shut down

operation will wait for active requests to complete. While the server is suspended, management requests are still processed.

Graceful shutdown is coordinated at a server-wide level, mostly focused on the entry points at which a request enters the server. The following subsystems support graceful shutdown:

Undertow

The **undertow** subsystem will wait for all requests to finish.

mod_cluster

The **modcluster** subsystem will notify the load balancer that the server is suspending in the **PRE_SUSPEND** phase.

EJB

The **ejb3** subsystem will wait for all remote EJB requests and MDB message deliveries to finish. Delivery to MDBs is stopped in the **PRE_SUSPEND** phase. EJB timers are suspended, and missed timers will be activated when the server is resumed.

Transactions

Once suspended, the server will not accept new requests, but in-flight transactions and requests are allowed to continue until they complete or until the timeout period expires. This also applies for web service requests associated with an XTS transaction.



NOTE

By default, transaction graceful shutdown is disabled for the **ejb** subsystem. You must enable transaction graceful shutdown if you want the server to wait for EJB-related transactions to complete before suspending. For example:

```
/subsystem=ejb3:write-attribute(name=enable-graceful-txn-shutdown,value=true)
```

EE Concurrency

The server will wait for all active jobs to finish. All queued jobs will be skipped. Currently, since EE Concurrency does not have persistence, those queued jobs that were skipped will be lost.

While the server is in a suspended state, scheduled tasks will continue to execute at their scheduled times but will throw a **java.lang.IllegalStateException**. Once the server is resumed, scheduled tasks will continue to execute normally, and in most cases tasks will not need to be rescheduled.

Batch

The server will stop all running jobs within the timeout period and defer all scheduled jobs.



NOTE

Graceful shutdown currently will not reject new inbound JMS messages. EE batch jobs and EE concurrency tasks scheduled by in-flight activity are currently allowed to proceed; however, EE concurrency tasks submitted that pass the timeout window currently error when executed.

Requests are tracked by the **request-controller** subsystem. Without this subsystem, suspend and resume capabilities are limited and the server will not wait for requests to complete before suspending or shutting down; however, if you do not need this capability, the **request-controller** subsystem can be removed for a small performance improvement.

2.4.1. Suspend Servers

JBoss EAP 7 introduced a *suspend* mode, which suspends server operations gracefully. This allows all active requests to complete normally, but will not accept any new requests. Once the server has been suspended, it can be shut down, returned back to a running state, or left in a suspended state to perform maintenance.



NOTE

The management interfaces are not impacted by suspending the server.

The server can be suspended and resumed using the management console or the management CLI.

Check the Server Suspend State

The server suspend state can be viewed using the following management CLI commands. The resulting value will be one of **RUNNING**, **PRE_SUSPEND**, **SUSPENDING**, or **SUSPENDED**.

- Check the suspend state for a standalone server.

```
:read-attribute(name=suspend-state)
```

- Check the suspend state for a server in a managed domain.

```
/host=master/server=server-one:read-attribute(name=suspend-state)
```

Suspend

Use the following management CLI commands to suspend the server, specifying the timeout value, in seconds, for the server to wait for active requests to complete. The default is **0**, which will suspend immediately. A value of **-1** will cause the server to wait indefinitely for all active requests to complete.

Each example waits up to 60 seconds for requests to complete before suspending.

- Suspend a standalone server.

```
:suspend(suspend-timeout=60)
```

- Suspend all servers in a managed domain.

```
:suspend-servers(suspend-timeout=60)
```

- Suspend a single server in a managed domain.

```
/host=master/server-config=server-one:suspend(suspend-timeout=60)
```

- Suspend all servers in a server group.

```
/server-group=main-server-group:suspend-servers(suspend-timeout=60)
```

- Suspend all servers at the host level.

```
/host=master:suspend-servers(suspend-timeout=60)
```

Resume

The **resume** command returns the server to a normal running state to accept new requests. You can initiate the command at the host, server, server group, or domain level. For example:

```
:resume
```

Start a Server in a Suspended State

You can start a server in a suspended state so that no requests are accepted by the server until it is resumed.

- To start a standalone server in a suspended state, use the **--start-mode=suspend** runtime argument when starting the JBoss EAP instance.

```
$ EAP_HOME/bin/standalone.sh --start-mode=suspend
```

- To start a managed domain server in a suspended state, pass the **start-mode=suspend** argument to the **start** operation in the management CLI command.

```
/host=HOST_NAME/server-config=SERVER_NAME:start(start-mode=suspend)
```



NOTE

You can also pass the **start-mode** argument to the **reload** and **restart** operations for a server.

- To start all servers in a managed domain server group in a suspended state, pass the **start-mode=suspend** argument to the **start-servers** operation in the management CLI command.

```
/server-group=SERVER_GROUP_NAME:start-servers(start-mode=suspend)
```



NOTE

You can also pass the **start-mode** argument to the **reload-servers** and **restart-servers** operations for a server group.

2.4.2. Shut Down Servers Gracefully Using the Management CLI

A server will be shut down gracefully if an appropriate timeout value is specified when stopping the server. Once the command is issued, the server will be suspended and will wait up to the specified timeout for all requests to finish before shutting down.

Use the following management CLI commands to shut down the server gracefully. Specify the timeout value, in seconds, for the server to wait for active requests to complete. The default is **0**, which will shut down the server immediately. A value of **-1** will cause the server to wait indefinitely for all active requests to complete before shutting down.

Each example waits up to 60 seconds for requests to complete before shutting down.

- Shut down a standalone server gracefully.

```
shutdown --suspend-timeout=60
```

- Stop all servers in a managed domain gracefully.

```
■
```



```
:stop-servers(suspend-timeout=60)
```

- Stop a single server in a managed domain gracefully.

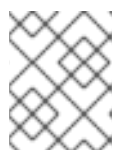
```
/host=master/server-config=server-one:stop(suspend-timeout=60)
```

- Stop all servers in a server group gracefully.

```
/server-group=main-server-group:stop-servers(suspend-timeout=60)
```

- Shutdown the host controller and all the servers it manages.

```
/host=master:shutdown(suspend-timeout=60)
```



NOTE

The **suspend-timeout** attribute is only applied to the servers managed by the host controller, not the host controller itself.

2.4.3. Shut Down Servers Gracefully Using an OS Signal

A server can be shut down gracefully by sending an OS **TERM** signal, such as with **kill -15 PID**. By default, this value is identical to the management CLI's **shutdown --suspend-timeout=0** command, resulting in immediate termination of any currently processing requests. The timeout can be configured by **org.wildfly.sigterm.suspend.timeout** system property, indicating the maximum number of seconds to wait for requests to complete before the server shuts down. A value of **-1** indicates that the server will wait indefinitely.



IMPORTANT

In a managed domain OS signals should not be used to shut down servers. Instead, servers should be [shut down using the management CLI](#) and through the managing Host Controller.

Graceful shutdown using an OS signal will not work if the JVM is configured to disable signal handling, such as if the **-Xrs** java argument has been passed to the JVM options, or if the signal sent is not one the process can respond to, such as if a **KILL** signal is sent.

2.5. STARTING AND STOPPING JBOSS EAP (RPM INSTALLATION)

Starting and stopping JBoss EAP is different for an RPM installation compared to a ZIP or installer installation.

2.5.1. Starting JBoss EAP (RPM Installation)

The command for starting an RPM installation of JBoss EAP depends on which operating mode you want to start, a standalone server or a managed domain, and which Red Hat Enterprise Linux version you are running.

Start JBoss EAP as a Standalone Server (RPM Installation)

- For Red Hat Enterprise Linux 6:

—

```
$ service eap7-standalone start
```

- For Red Hat Enterprise Linux 7 and later:

```
$ systemctl start eap7-standalone.service
```

This will start JBoss EAP using the **standalone.xml** configuration file by default. You can start JBoss EAP with a different [standalone server configuration file](#) by setting a property in the [RPM service configuration file](#). For more information, see the [Configure RPM Service Properties](#) section below.

Start JBoss EAP in a Managed Domain (RPM Installation)

- For Red Hat Enterprise Linux 6:

```
$ service eap7-domain start
```

- For Red Hat Enterprise Linux 7 and later:

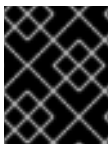
```
$ systemctl start eap7-domain.service
```

This will start JBoss EAP using the **host.xml** configuration file by default. You can start JBoss EAP with a different [managed domain configuration file](#) by setting a property in the [RPM service configuration file](#). For more information, see the [Configure RPM Service Properties](#) section below.

Configure RPM Service Properties

This section shows you how to configure the RPM service properties and other startup options for your JBoss EAP installation. Note that it is recommended to back up your configuration files before making modifications.

For a listing of all available startup options for an RPM installation, see the [RPM Service Configuration Properties](#) section.



IMPORTANT

For Red Hat Enterprise Linux 7 and later, RPM service configuration files are loaded using **systemd**, so variable expressions are not expanded.

- Specify the server configuration file.
When starting a standalone server, the **standalone.xml** file is used by default. When running in a managed domain, the **host.xml** file is used by default. You can start JBoss EAP with a different configuration file by setting the **WILDFLY_SERVER_CONFIG** property in the appropriate [RPM configuration file](#), for example, **eap7-standalone.conf**.

```
WILDFLY_SERVER_CONFIG=standalone-full.xml
```

- Bind to a specific IP address.
By default, a JBoss EAP RPM installation binds to **0.0.0.0**. You can bind JBoss EAP to a specific IP address by setting the **WILDFLY_BIND** property in the appropriate [RPM configuration file](#), for example, **eap7-standalone.conf**.

```
WILDFLY_BIND=192.168.0.1
```

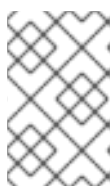
**NOTE**

If you want to bind the management interfaces to a specific IP address, this can be configured in the JBoss EAP startup configuration file as shown in the next example.

- Set JVM options or Java properties.

You can specify JVM options or Java properties to pass into the JBoss EAP startup script by editing the startup configuration file. This file is **EAP_HOME/bin/standalone.conf** for a standalone server or **EAP_HOME/bin/domain.conf** for a managed domain. The below example configures the heap size and binds the JBoss EAP management interfaces to an IP address.

```
JAVA_OPTS="$JAVA_OPTS -Xms2048m -Xmx2048m"
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address.management=192.168.0.1"
```

**NOTE**

If required, the JBoss EAP bind address must be configured using the **WILDFLY_BIND** property and not using the **jboss.bind.address** standard property here.

**NOTE**

If a property has the same name in both the RPM service configuration file, such as **/etc/sysconfig/eap7-standalone**, and in the JBoss EAP startup configuration file, such as **EAP_HOME/bin/standalone.conf**, the value that takes precedence is the one in the JBoss EAP startup configuration file. One such property is **JAVA_HOME**.

2.5.2. Stopping JBoss EAP (RPM Installation)

The command for stopping an RPM installation of JBoss EAP depends on which operating mode that was started, a standalone server or a managed domain, and which Red Hat Enterprise Linux version you are running.

Stop JBoss EAP as a Standalone Server (RPM Installation)

- For Red Hat Enterprise Linux 6:

```
$ service eap7-standalone stop
```

- For Red Hat Enterprise Linux 7 and later:

```
$ systemctl stop eap7-standalone.service
```

Stop JBoss EAP in a Managed Domain (RPM Installation)

- For Red Hat Enterprise Linux 6:

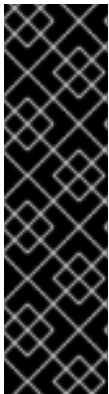
```
$ service eap7-domain stop
```

- For Red Hat Enterprise Linux 7 and later:

```
$ systemctl stop eap7-domain.service
```

For a listing of all available startup options for an RPM installation, see the [RPM Service Configuration Files](#) section.

2.6. POWERSHELL SCRIPTS (WINDOWS SERVER)



IMPORTANT

The collection of PowerShell scripts is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

JBoss EAP includes PowerShell script equivalents for most of the JBoss EAP management scripts. This includes a PowerShell script to start JBoss EAP on Microsoft Windows Server.

The JBoss EAP PowerShell scripts are designed to work with PowerShell version 2 and newer running on tested versions of Windows Server.

The JBoss EAP PowerShell scripts are located in ***EAP_HOME*bin**, and are used in mostly the same way as the JBoss EAP batch scripts.

For example, to start a standalone JBoss EAP server with the **standalone-full.xml** configuration file, use the following PowerShell command:

```
.\standalone.ps1 "-c=standalone-full.xml"
```



NOTE

Arguments of the JBoss EAP PowerShell scripts must be in quotes.

CHAPTER 3. JBOSS EAP MANAGEMENT

JBoss EAP uses a simplified configuration, with one configuration file per standalone server or managed domain. Default configuration for a standalone server is stored in the ***EAP_HOME/standalone/configuration/standalone.xml*** file and default configuration for a managed domain is stored in the ***EAP_HOME/domain/configuration/domain.xml*** file. Additionally, the default configuration for a host controller is stored in the ***EAP_HOME/domain/configuration/host.xml*** file.

JBoss EAP can be configured using the command-line management CLI, web-based management console, Java API, or HTTP API. Changes made using these management interfaces persist automatically, and the XML configuration files are overwritten by the Management API. The management CLI and management console are the preferred methods, and it is not recommended to edit the XML configuration files manually.

3.1. ABOUT SUBSYSTEMS, EXTENSIONS, AND PROFILES

Different aspects of JBoss EAP functionality are configured in different subsystems. For example, application and server logging are configured in the **logging** subsystem.

An *extension* is a module that extends the core functionality of the server. Extensions are loaded as they are needed by deployments, and are unloaded when they are no longer needed. See the JBoss EAP *Management CLI Guide* for how to [add](#) and [remove](#) extensions.

A *subsystem* provides configuration options for a particular extension. For more information on the available subsystems, see [Overview of JBoss EAP Subsystems](#).

A collection of subsystem configurations makes up a *profile*, which is configured to satisfy the needs for the server. A standalone server has a single, unnamed profile. A managed domain can define many [profiles](#) for use by server groups in the domain.

Using the Management Console or the Management CLI

Both the management console and the management CLI are valid, supported ways of updating the configuration of a JBoss EAP instance. Deciding between the two is a matter of preference. Those who prefer to use a graphical, web-based interface should use the management console. Those who prefer a command-line interface should use the management CLI.

3.2. MANAGEMENT USERS

The default JBoss EAP configuration provides local authentication so that a user can access the management CLI on the local host without requiring authentication.

However, you must add a management user if you want to access the management CLI remotely or use the management console, which is considered remote access even if the traffic originates on the local host. If you attempt to access the management console before adding a management user, you will receive an error message.

If JBoss EAP is installed using the graphical installer, then a management user is created during the installation process.

This guide covers simple user management for JBoss EAP using the **add-user** script, which is a utility for adding new users to the properties files for out-of-the-box authentication.

For more advanced authentication and authorization options, such as LDAP or Role-Based Access Control (RBAC), see the [Core Management Authentication](#) section of the JBoss EAP *Security Architecture*.

3.2.1. Adding a Management User

1. Run the **add-user** utility script and follow the prompts.

```
$ EAP_HOME/bin/add-user.sh
```



NOTE

For Windows Server, use the **EAP_HOME\bin\add-user.bat** script.

2. Press **ENTER** to select the default option **a** to add a management user.
This user will be added to the *ManagementRealm* and will be authorized to perform management operations using the management console or management CLI. The other choice, **b**, adds a user to the *ApplicationRealm*, which is used for applications and provides no particular permissions.
3. Enter the desired username and password. You will be prompted to confirm the password.



NOTE

User names can only contain the following characters, in any number and in any order:

- Alphanumeric characters (a-z, A-Z, 0-9)
- Dashes (-), periods (.), commas (,), at sign (@)
- Backslash (\)
- Equals (=)

By default, JBoss EAP allows weak passwords but will issue a warning.

See [Setting Add-User Utility Password Restrictions](#) for details on changing this default behavior.

4. Enter a comma-separated list of groups to which the user belongs. If you do not want the user to belong to any groups, press **ENTER** to leave it blank.
5. Review the information and enter **yes** to confirm.
6. Determine whether this user represents a remote JBoss EAP server instance. For a basic management user, enter **no**.
One type of user that may need to be added to the *ManagementRealm* is a user representing another instance of JBoss EAP, which must be able to authenticate to join as a member of a cluster. If this is the case, then answer **yes** to this prompt and you will be given a hashed secret value representing the user's password, which will need to be added to a different configuration file.

Users can also be created non-interactively by passing parameters to the **add-user** script. This approach is not recommended on shared systems, because the passwords will be visible in log and history files. For more information, see [Running the Add-User Utility Non-Interactively](#).

3.2.2. Running the Add-User Utility Non-Interactively

You can run the **add-user** script non-interactively by passing in arguments on the command line. At a minimum, the username and password must be provided.



WARNING

This approach is not recommended on shared systems, because the passwords will be visible in log and history files.

Create a User Belonging to Multiple Groups

The following command adds a management user, **mgmtuser1**, with the **guest** and **mgmtgroup** groups.

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser1' -p 'password1!' -g 'guest,mgmtgroup'
```

Specify an Alternative Properties File

By default, user and group information created using the **add-user** script are stored in properties files located in the server configuration directory.

User information is stored in the following properties files:

- **EAP_HOME/standalone/configuration/mgmt-users.properties**
- **EAP_HOME/domain/configuration/mgmt-users.properties**

Group information is stored in the following properties files:

- **EAP_HOME/standalone/configuration/mgmt-groups.properties**
- **EAP_HOME/domain/configuration/mgmt-groups.properties**

These default directories and properties file names can be overridden. The following command adds a new user, specifying a different name and location for the user properties files.

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser2' -p 'password1!' -sc '/path/to/standaloneconfig' -dc  
'/path/to/domainconfig' -up 'newname.properties'
```

The new user was added to the user properties files located at **/path/to/standaloneconfig/newname.properties** and **/path/to/domainconfig/newname.properties**. Note that these files must already exist or you will see an error.

For a complete listing of all available **add-user** arguments and their purposes, use the **--help** argument or see the [Add-User Utility Arguments](#) section.

3.2.3. Add-User Utility Password Restrictions

The password restrictions for the **add-user** utility script can be configured using the **EAP_HOME/bin/add-user.properties** file.



IMPORTANT

The **add-user.properties** file is an unprotected plain-text file and must be secured to avoid unwarranted access to its contents.

To avoid setting an unintentional password, check that your keyboard's system keymap is correct. The default system keymap is **en-qwerty**. If you change this default setting and create a new password, you must check that the password meets the criteria located in the class

SimplePasswordStrengthChecker.

By default, JBoss EAP allows weak passwords but issues a warning. To reject passwords that do not meet the minimum requirements specified, set the **password.restriction** property to **REJECT**.

The following table describes the additional password requirement settings that can be configured in the **EAP_HOME/bin/add-user.properties** file:

Table 3.1. Additional Password Requirement Settings

Attribute	Description
minLength	The minimum number of characters for a password. For example, password.restriction.minLength=8 restricts the password to a minimum of eight characters.
strength	<p>Sets the threshold that a password must meet to be valid. Valid threshold entries include:</p> <ul style="list-style-type: none"> * VERY_WEAK * WEAK * MODERATE * MEDIUM * STRONG * VERY_STRONG * EXCEPTIONAL <p>The default value is MODERATE. The threshold values are defined and the default value specified in the class SimplePasswordStrengthChecker.</p> <p>NOTE: If you do not specify a threshold value, MODERATE becomes the default value. This value is specified in the class SimplePasswordStrengthChecker.</p>
minAlpha	The minimum number of alphabetic characters set for a password. For example, password.restriction.minAlpha=1 restricts a password to include at least one alphabetic character.
minDigit	The minimum number of numeric characters set for a password. For example, password.restriction.minDigit=1 restricts a password to include at least one numerical character.

Attribute	Description
minSymbol	The minimum number of symbols set for a password. For example, password.restriction.minSymbol=1 restricts a password to include at least one symbol.
forbiddenValue	Restricts a user from setting an easily determined password, such as <i>root</i> . For example, password.restriction.forbidden=root,admin,administrator restricts setting <i>root</i> , <i>admin</i> , or <i>administrator</i> as the password.
mustNotMatchUsername	Restricts the user from setting their user name as the password. For example, password.restriction.mustNotMatchUsername=TRUE restricts the user from setting their user name as the password. This rule is ignored if set to false .

Additional Resources

See the [Configuring Basic System Settings](#) guide on the Red Hat Customer Portal.

3.2.4. Updating a Management User

You can update the settings for an existing management user using the **add-user** utility script by entering the username when prompted.

```
$ EAP_HOME/bin/add-user.sh
```

```
What type of user do you wish to add?
```

- ```
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
```

```
(a): a
```

```
Enter the details of the new user to add.
```

```
Using realm 'ManagementRealm' as discovered from the existing property files.
```

```
Username : test-user
```

```
User 'test-user' already exists and is enabled, would you like to...
```

- ```
a) Update the existing user password and roles
b) Disable the existing user
c) Type a new username
```

```
(a):
```

When you enter a username that already exists, you are presented with several options:

- Type **a** to update the password for the existing user.
- Type **b** to disable the existing user.
- Type **c** to enter a new username.

**WARNING**

When updating a user using the **add-user** script non-interactively, the user is updated automatically with no confirmation prompt.

3.3. OPTIMIZING THE JBOSS EAP SERVER CONFIGURATION

Once you have [installed the JBoss EAP server](#), and you have [created a management user](#), Red Hat recommends that you optimize your server configuration.

Make sure you review information in the [Performance Tuning Guide](#) for information about how to optimize the server configuration to avoid common problems when deploying applications in a production environment. Common optimizations include [setting ulimits](#), [enabling garbage collection](#), [creating Java heap dumps](#), and [adjusting the thread pool size](#).

It is also a good idea to apply any existing patches for your release of the product. Each patch for EAP contains numerous bug fixes. For more information, see [Patching JBoss EAP](#) in the *Patching and Upgrading Guide* for JBoss EAP.

3.4. MANAGEMENT INTERFACES

3.4.1. Management CLI

The management command-line interface (CLI) is a command-line administration tool for JBoss EAP.

Use the management CLI to start and stop servers, deploy and undeploy applications, configure system settings, and perform other administrative tasks. Operations can be performed in batch mode, allowing multiple tasks to be run as a group.

Many common terminal commands are available, such as **ls**, **cd**, and **pwd**. The management CLI also supports tab completion.

For detailed information on using the management CLI, including commands and operations, syntax, and running in batch mode, see the JBoss EAP [Management CLI Guide](#).

Launch the Management CLI

```
$ EAP_HOME/bin/jboss-cli.sh
```

**NOTE**

For Windows Server, use the **EAP_HOME\bin\jboss-cli.bat** script.

Connect to a Running Server

```
connect
```

Or you can launch the management CLI and connect in one step by using the **EAP_HOME/bin/jboss-cli.sh --connect** command.

Display Help

Use the following command for general help.

```
help
```

Use the **--help** flag on a command to receive instructions on using that specific command. For instance, to receive information on using **deploy**, the following command is executed.

```
deploy --help
```

Quit the Management CLI

```
quit
```

View System Settings

The following command uses the **read-attribute** operation to display whether the example datasource is enabled.

```
/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
{
  "outcome" => "success",
  "result" => true
}
```

When running in a managed domain, you must specify which profile to update by preceding the command with **/profile=PROFILE_NAME**.

```
/profile=default/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
```

Update System Settings

The following command uses the **write-attribute** operation to disable the example datasource.

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled,value=false)
```

Start Servers

The management CLI can also be used to start and stop servers when running in a managed domain.

```
/host=HOST_NAME/server-config=server-one:start
```

3.4.2. Management Console

The management console is a web-based administration tool for JBoss EAP.

Use the management console to start and stop servers, deploy and undeploy applications, tune system settings, and make persistent modifications to the server configuration. The management console also has the ability to perform administrative tasks, with live notifications when any changes performed by the current user require the server instance to be restarted or reloaded.

In a managed domain, server instances and server groups in the same domain can be centrally managed from the management console of the domain controller.

For a JBoss EAP instance running on the local host using the default management port, the management console can be accessed through a web browser at

<http://localhost:9990/console/index.html>. You will need to authenticate with a user that has permissions to access the management console.

The management console provides the following tabs for navigating and managing your JBoss EAP standalone server or managed domain.

Home

Learn how to accomplish several common configuration and management tasks. Take a tour to become familiar with the JBoss EAP management console.

Deployments

Add, remove, and enable deployments. In a managed domain, assign deployments to server groups.

Configuration

Configure available subsystems, which provide capabilities such as web services, messaging, or high availability. In a managed domain, manage the profiles that contain different subsystem configurations.

Runtime

View runtime information, such as server status, JVM usage, and server logs. In a managed domain, manage your hosts, server groups, and servers.

Patching

Apply patches to your JBoss EAP instances.

Access Control

Assign roles to users and groups when using Role-Based Access Control.

3.4.2.1. Updating Attributes in the Management Console

Once you have navigated to the appropriate section of the management console for the resource that you want to modify, you can edit its attributes as long as you have the proper permissions.

1. Click the **Edit** link.
2. Make the desired changes.
Required fields are marked with an asterisk (*). You can view attribute descriptions by clicking the **Help** link.



NOTE

Depending on the attribute type, the input field can be a text field, ON/OFF field, or drop down. In some text fields, as you type, values from elsewhere in the configuration may appear as suggestions.

3. Click **Save** to save the changes.
4. If necessary, reload the server for the changes to take effect.
A popup appears when saving changes that require a reload in order to take effect. To reload a standalone server, click the **Reload** link in the popup. To reload a server in a managed domain, click the **Topology** link, select the appropriate server, and click the **Reload** drop down option.

To view the history of recent configuration actions you have performed, click the notification icon in the top-right of the management console.

3.4.2.2. Enable/Disable Management Console

You can enable or disable the management console by setting the **console-enabled** boolean attribute of the **/core-service=management/management-interface=http-interface** resource. For the master host in domain mode, use **/host=master/core-service=management/management-interface=http-interface**.

For example, to enable:

```
/core-service=management/management-interface=http-interface:write-attribute(name=console-enabled,value=true)
```

For example, to disable:

```
/core-service=management/management-interface=http-interface:write-attribute(name=console-enabled,value=false)
```

3.4.2.3. Change the Language of the Management Console

By default, the language settings of the management console is English. You can choose to use one of the following languages instead:

- German (de)
- Simplified Chinese (zh-Hans)
- Brazilian Portuguese (pt-BR)
- French (fr)
- Spanish (es)
- Japanese (ja)

To Change the Language of the Management Console

1. Log in to the management console.
2. Click the **Settings** link in the lower-right corner of the management console.
3. Select the required language from the **Locale** selection box.
4. Select **Save**. A confirmation box informs you that you need to reload the application.
5. Click **Yes**. The system refreshes your web browser automatically to use the selected locale.

3.4.2.4. Customizing the Management Console Title

You can customize the management console title so that each of your JBoss EAP instances can be identified at a quick glance.

To customize the management console title:

1. Log in to the management console.
2. Click **Settings** in the lower-right corner of the management console.
3. In the **Settings** window, modify the title in the **Title** field.

4. Click **Save**.

A confirmation box informs you that you must reload the management console.

5. Click **Yes**.

The system refreshes your web browser automatically and the new title is displayed on the tab header.

3.5. MANAGEMENT APIS

3.5.1. HTTP API

The HTTP API endpoint is the entry point for management clients that rely on the HTTP protocol to integrate with the JBoss EAP management layer.

The HTTP API is used by the JBoss EAP management console but offers integration capabilities for other clients as well. By default, the HTTP API is accessible at **`http://HOST_NAME:9990/management`**. This URL will display the raw attributes and values exposed to the API.

Read Resources

While you can read, write, or perform other operations using the HTTP **POST** method, you can perform some read operations using a **GET** request. The HTTP **GET** method uses the following URL format.

```
http://HOST_NAME:9990/management/PATH_TO_RESOURCE?
operation=OPERATION&PARAMETER=VALUE
```

Be sure to replace all of the replaceable values with those that are appropriate for your request. The following values are the available options for the **OPERATION** replaceable value:

Value	Description
attribute	Performs the read-attribute operation.
operation-description	Performs the read-operation-description operation.
operation-names	Performs the read-operation-names operation.
resource	Performs the read-resource operation.
resource-description	Performs the read-resource-description operation.
snapshots	Performs the list-snapshots operation.

The following example URLs show how to perform read operations using the HTTP API.

Example: Read All Attributes and Values for a Resource

```
http://HOST_NAME:9990/management/subsystem/undertow/server/default-server/http-
listener/default
```

This displays all attributes and their values for the **default** HTTP listener.

**NOTE**

The default operation is **read-resource**.

Example: Read the Value of an Attribute for a Resource

```
http://HOST_NAME:9990/management/subsystem/datasources/data-source/ExampleDS?
operation=attribute&name=enabled
```

This reads the value of the **enabled** attribute for the **ExampleDS** datasource.

Update Resources

You can use the HTTP **POST** method to update configuration values or perform other operations using the HTTP API. You must provide authentication for these operations.

The following examples show how to update resources using the HTTP API.

Example: Update the Value of an Attribute for a Resource

```
$ curl --digest http://HOST_NAME:9990/management --header "Content-Type: application/json" -u
USERNAME:PASSWORD -d '{"operation": "write-attribute", "address":
["subsystem", "datasources", "data-source", "ExampleDS"], "name": "enabled", "value": "false",
"json.pretty": "1"}'
```

This updates the value of the **enabled** attribute for the **ExampleDS** datasource to **false**.

Example: Issue an Operation to the Server

```
$ curl --digest http://localhost:9990/management --header "Content-Type: application/json" -u
USERNAME:PASSWORD -d '{"operation": "reload"}'
```

This reloads the server.

See [Deploying Applications Using the HTTP API](#) for information on how to deploy applications to JBoss EAP using the HTTP API.

3.5.2. Native API

The native API endpoint is the entry point for management clients that rely on the native protocol to integrate with the JBoss EAP management layer. The native API is used by the JBoss EAP management CLI but offers integration capabilities for other clients as well.

The following Java code shows an example of how to execute management operations from Java code using the native API.

**NOTE**

You must add the required JBoss EAP libraries, found in the **EAP_HOME/bin/client/jboss-cli-client.jar** file, to your class path.

Example: Using the Native API to Read Resources

```
// Create the management client
```

```

ModelControllerClient client = ModelControllerClient.Factory.create("localhost", 9990);

// Create the operation request
ModelNode op = new ModelNode();

// Set the operation
op.get("operation").set("read-resource");

// Set the address
ModelNode address = op.get("address");
address.add("subsystem", "undertow");
address.add("server", "default-server");
address.add("http-listener", "default");

// Execute the operation and manipulate the result
ModelNode returnVal = client.execute(op);
System.out.println("Outcome: " + returnVal.get("outcome").toString());
System.out.println("Result: " + returnVal.get("result").toString());

// Close the client
client.close();

```

3.6. CONFIGURATION DATA

3.6.1. Standalone Server Configuration Files

The standalone configuration files are located in the ***EAP_HOME/standalone/configuration/*** directory. A separate file exists for each of the five predefined profiles (*default*, *ha*, *full*, *full-ha*, *load-balancer*).

Table 3.2. Standalone Configuration Files

Configuration File	Purpose
standalone.xml	This standalone configuration file is the default configuration that is used when you start your standalone server. It contains all information about the server, including subsystems, networking, deployments, socket bindings, and other configurable details. It does not provide the subsystems necessary for messaging or high availability.
standalone-ha.xml	This standalone configuration file includes all of the default subsystems and adds the modcluster and jgroups subsystems for high availability. It does not provide the subsystems necessary for messaging.
standalone-full.xml	This standalone configuration file includes all of the default subsystems and adds the messaging-activemq and iiop-openjdk subsystems. It does not provide the subsystems necessary for high availability.
standalone-full-ha.xml	This standalone configuration file includes support for every possible subsystem, including those for messaging and high availability.

Configuration File	Purpose
standalone-load-balancer.xml	This standalone configuration file includes the minimum subsystems necessary to use the built-in <code>mod_cluster</code> front-end load balancer to load balance other JBoss EAP instances.

By default, starting JBoss EAP as a standalone server uses the **standalone.xml** file. To start JBoss EAP with a different configuration, use the **--server-config** argument. For example,

```
$ EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml
```

3.6.2. Managed Domain Configuration Files

The managed domain configuration files are located in the **EAP_HOME/domain/configuration/** directory.

Table 3.3. Managed Domain Configuration Files

Configuration File	Purpose
domain.xml	This is the main configuration file for a managed domain. Only the domain master reads this file. This file contains the configurations for all of the profiles (<i>default, ha, full, full-ha, load-balancer</i>).
host.xml	This file includes configuration details specific to a physical host in a managed domain, such as network interfaces, socket bindings, the name of the host, and other host-specific details. The host.xml file includes all of the features of both host-master.xml and host-slave.xml , which are described below.
host-master.xml	This file includes only the configuration details necessary to run a server as the master domain controller.
host-slave.xml	This file includes only the configuration details necessary to run a server as a managed domain host controller.

By default, starting JBoss EAP in a managed domain uses the **host.xml** file. To start JBoss EAP with a different configuration, use the **--host-config** argument. For example,

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml
```

3.6.3. Backing Up Configuration Data

In order to later restore the JBoss EAP server configuration, items in the following locations should be backed up:

- **EAP_HOME/standalone/configuration/**
 - Back up the entire directory to save user data, server configuration, and logging settings for standalone servers.

- ***EAP_HOME*/domain/configuration/**
 - Back up the entire directory to save user and profile data, domain and host configuration, and logging settings for managed domains.
- ***EAP_HOME*/modules/**
 - Back up any custom modules.
- ***EAP_HOME*/welcome-content/**
 - Back up any custom welcome content.
- ***EAP_HOME*/bin/**
 - Back up any custom scripts or startup configuration files.

3.6.4. Configuration File Snapshots

To assist in the maintenance and management of the server, JBoss EAP creates a timestamped version of the original configuration file at the time of startup. Any additional configuration changes made by management operations will result in the original file being automatically backed up, and a working copy of the instance being preserved for reference and rollback. Additionally, configuration snapshots can be taken, which are point-in-time copies of the current server configuration. These snapshots can be saved and loaded by an administrator.

The following examples use the **standalone.xml** file, but the same process applies to the **domain.xml** and **host.xml** files.

Take a Snapshot

Use the management CLI to take a snapshot of the current configurations.

```
:take-snapshot
{
  "outcome" => "success",
  "result" => "EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20151022-133109702standalone.xml"
}
```

List Snapshots

Use the management CLI to list all snapshots that have been taken.

```
:list-snapshots
{
  "outcome" => "success",
  "result" => {
    "directory" => "EAP_HOME/standalone/configuration/standalone_xml_history/snapshot",
    "names" => [
      "20151022-133109702standalone.xml",
      "20151022-132715958standalone.xml"
    ]
  }
}
```

Delete a Snapshot

Use the management CLI to delete a snapshot.

```
:delete-snapshot(name=20151022-133109702standalone.xml)
```

Start the Server with a Snapshot

The server can be started using a snapshot or an automatically-saved version of the configuration.

1. Navigate to the ***EAP_HOME/standalone/configuration/standalone_xml_history*** directory and identify the snapshot or saved configuration file to be loaded.
2. Start the server and point to the selected configuration file. Pass in the file path relative to the configuration directory, ***EAP_HOME/standalone/configuration/***.

```
$ EAP_HOME/bin/standalone.sh --server-  
config=standalone_xml_history/snapshot/20151022-133109702standalone.xml
```

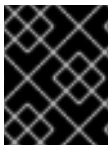


NOTE

When running in a managed domain, use the ***--host-config*** argument instead to specify the configuration file.

3.6.5. View Configuration Changes

JBoss EAP 7 provides the ability to track configuration changes made to the running system. This allows administrators to view a history of configuration changes made by other authorized users.



IMPORTANT

Changes are stored in memory and are not persisted between server restarts. This feature is not a replacement for [management audit logging](#).

You can enable tracking and view configuration changes from either the [management CLI](#) or the [management console](#).

Track and View Configuration Changes from the Management CLI

To enable tracking of configuration changes, use the following management CLI command. You can specify how many entries to store using the ***max-history*** attribute.

```
/subsystem=core-management/service=configuration-changes:add(max-history=20)
```



NOTE

In a managed domain, configuration changes are tracked at the host level for host and server-related modifications. Enabling configuration changes for a host controller enables it for all of its managed servers. You can track configuration changes per host using the following command.

```
/host=HOST_NAME/subsystem=core-management/service=configuration-  
changes:add(max-history=20)
```

To view the list of most recent configuration changes, use the following management CLI command.

```
/subsystem=core-management/service=configuration-changes:list-changes
```



NOTE

In a managed domain, you can list the configuration changes for a host using the following command.

```
/host=HOST_NAME/subsystem=core-management/service=configuration-
changes:list-changes
```

You can list the configuration changes that affect a particular server using the following command.

```
/host=HOST_NAME/server=SERVER_NAME/subsystem=core-
management/service=configuration-changes:list-changes
```

This lists each configuration change made, with the date, origin, outcome, and operation details. For example, the below output from the **list-changes** command shows configuration changes, with the most recent displayed first.

```
{
  "outcome" => "success",
  "result" => [
    {
      "operation-date" => "2016-02-12T18:37:00.354Z",
      "access-mechanism" => "NATIVE",
      "remote-address" => "127.0.0.1/127.0.0.1",
      "outcome" => "success",
      "operations" => [{
        "address" => [],
        "operation" => "reload",
        "operation-headers" => {
          "caller-type" => "user",
          "access-mechanism" => "NATIVE"
        }
      }]
    },
    {
      "operation-date" => "2016-02-12T18:34:16.859Z",
      "access-mechanism" => "NATIVE",
      "remote-address" => "127.0.0.1/127.0.0.1",
      "outcome" => "success",
      "operations" => [{
        "address" => [
          ("subsystem" => "datasources"),
          ("data-source" => "ExampleDS")
        ],
        "operation" => "write-attribute",
        "name" => "enabled",
        "value" => false,
        "operation-headers" => {
          "caller-type" => "user",
          "access-mechanism" => "NATIVE"
        }
      }]
    }
  ],
}
```

```

    {
      "operation-date" => "2016-02-12T18:24:11.670Z",
      "access-mechanism" => "HTTP",
      "remote-address" => "127.0.0.1/127.0.0.1",
      "outcome" => "success",
      "operations" => [{
        "operation" => "remove",
        "address" => [
          ("subsystem" => "messaging-activemq"),
          ("server" => "default"),
          ("jms-queue" => "ExpiryQueue")
        ],
        "operation-headers" => {"access-mechanism" => "HTTP"}
      ]
    }
  ]
}

```

This example lists the details of three operations performed that impacted the configuration:

- Reloading the server from the management CLI.
- Disabling the **ExampleDS** datasource from the management CLI.
- Removing the **ExpiryQueue** queue from the management console.

Track and View Configuration Changes from the Management Console

To enable tracking of configuration changes from the management console, select to the **Runtime** tab, navigate to the server or host to track changes for and select **Configuration Changes** from the drop down. Click **Enable Configuration Changes** and provide a maximum history value.

The table on this page then lists each configuration change made, with the date, origin, outcome, and operation details.

3.6.6. Property Replacement

JBoss EAP allows you to use expressions to define replaceable properties in place of literal values in the configuration. Expressions use the format **`${PARAMETER:DEFAULT_VALUE}`**. If the specified parameter is set, then the parameter's value will be used. Otherwise, the default value provided will be used.

The supported sources for resolving expressions are system properties, environment variables, and the vault. For deployments only, the source can be properties listed in a **META-INF/jboss.properties** file in the deployment archive. For deployment types that support subdeployments, the resolution is scoped to all subdeployments if the properties file is in the outer deployment, for example the EAR. If the properties file is in the subdeployment, then the resolution is scoped just to that subdeployment.

The example below from the **standalone.xml** configuration file sets the **inet-address** for the **public** interface to **127.0.0.1** unless the **jboss.bind.address** parameter is set.

```

<interface name="public">
  <inet-address value="${jboss.bind.address:127.0.0.1}"/>
</interface>

```

The **jboss.bind.address** parameter can be set when starting EAP as a standalone server with the following command:

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

Nested Expressions

Expressions can be nested, which allows for more advanced use of expressions in place of fixed values. The format of a nested expression is like that of a normal expression, but one expression is embedded in the other, for example:

```
${SYSTEM_VALUE_1}${SYSTEM_VALUE_2}}
```

Nested expressions are evaluated recursively, so the *inner* expression is first evaluated, then the *outer* expression is evaluated. Expressions may also be recursive, where an expression resolves to another expression, which is then resolved. Nested expressions are permitted anywhere that expressions are permitted, with the exception of management CLI commands.

An example of where a nested expression might be used is if the password used in a datasource definition is masked. The configuration for the datasource might have the following line:

```
<password>${VAULT::ds_ExampleDS::password::1}</password>
```

The value of **ds_ExampleDS** could be replaced with a system property (**datasource_name**) using a nested expression. The configuration for the datasource could instead have the following line:

```
<password>${VAULT::${datasource_name}::password::1}</password>
```

JBoss EAP would first evaluate the expression **\${datasource_name}**, then input this to the larger expression and evaluate the resulting expression. The advantage of this configuration is that the name of the datasource is abstracted from the fixed configuration.

Descriptor-Based Property Replacement

Application configuration, such as datasource connection parameters, typically varies between development, testing, and production environments. This variance is sometimes accommodated by build system scripts, as the Jakarta EE specification does not contain a method to externalize these configurations. With JBoss EAP, you can use descriptor-based property replacement to manage configuration externally.

Descriptor-based property replacement substitutes properties based on descriptors, allowing you to remove assumptions about the environment from the application and the build chain. Environment-specific configurations can be specified in deployment descriptors rather than annotations or build system scripts. You can provide configuration in files or as parameters at the command line.

There are several flags in the **ee** subsystem that control whether property replacement is applied.

JBoss-specific descriptor replacement is controlled by the **jboss-descriptor-property-replacement** flag and is *enabled* by default. When enabled, properties can be replaced in the following deployment descriptors:

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- ***-jms.xml**
- ***-ds.xml**

The following management CLI command can be used to enable or disable property replacement in JBoss-specific descriptors:

```
/subsystem=ee:write-attribute(name="jboss-descriptor-property-replacement",value=VALUE)
```

Jakarta EE descriptor replacement controlled by the **spec-descriptor-property-replacement** flag and is *disabled* by default. When enabled, properties can be replaced in the following deployment descriptors:

- **ejb-jar.xml**
- **persistence.xml**
- **application.xml**
- **web.xml**

The following management CLI command can be used to enable or disable property replacement in Jakarta EE descriptors:

```
/subsystem=ee:write-attribute(name="spec-descriptor-property-replacement",value=VALUE)
```

3.6.7. Using Git to Manage Configuration Data

As of JBoss EAP 7.3, you can use Git to manage and persist your server configuration data, properties files, and deployments. This not only allows you to manage the version history for these files, but it also allows you to share server and application configurations across multiple servers and nodes using one or more Git repositories. This feature only works for standalone servers that use the default configuration directory layout.

You can choose to use configuration data in a [local Git repository](#) or you can pull the data from a [remote Git repository](#). The Git repository is configured in the **jboss.server.base.dir** directory, which is the base directory for standalone server content. Once the **jboss.server.base.dir** directory is configured to use Git, JBoss EAP automatically commits every update you make to the configuration using the management CLI or management console. Any changes made outside of the server by manually editing the configuration files are not committed or persisted; however, you can use the Git CLI to add and commit manual changes. You can also use the Git CLI to view the commit history, manage branching, and manage the content.

To use this feature, pass one or more of the following arguments on the command line when you start the server.

Table 3.4. Server Startup Arguments for Git Configuration Management

Argument	Description
<code>--git-repo</code>	The location of the Git repository that is used to manage and persist server configuration data. This can be local if you want to store it locally, or the URL to a remote repository.

Argument	Description
<code>--git-branch</code>	The branch or tag name in the Git repository to use. This argument should name an existing branch or tag name as it will <i>not</i> be created if it does not exist. If you use a tag name, you put the repository in a detached HEAD state, meaning future commits are not attached to any branches. Tag names are read-only and are normally used when you need to replicate a configuration across several nodes.
<code>--git-auth</code>	The URL to an Elytron configuration file that contains the credentials to be used when connecting to a remote Git repository. This argument is required if your remote Git repository requires authentication. Although Git supports SSH authentication, Elytron does not; therefore, it is currently only possible to specify credentials to authenticate with HTTP or HTTPS, not SSH. This argument is not used with a local repository.

Using a Local Git Repository

To use a local Git repository, start the server with the `--git-repo=local` argument. You can also specify an optional branch or tag name in the remote repository by adding the `--git-branch=GIT_BRANCH_NAME` argument when you start the server. This argument should name an existing branch or tag name as it will *not* be created if it does not exist. If you use a tag name, you put the repository in a detached HEAD state, meaning future commits are not attached to any branches.

The following is an example of a command to start the server using the **1.0.x** branch of the **local** repository.

```
$ EAP_HOME/bin/standalone.sh --git-repo=local --git-branch=1.0.x
```

If you start the server with the argument to use a **local** Git repository, JBoss EAP checks whether the **jboss.server.base.dir** directory is already configured for Git. If not, JBoss EAP creates and initializes the Git repository in the **jboss.server.base.dir** directory using the existing configuration content. JBoss EAP checks out a branch name passed by the `--git-branch` argument. If that argument is not passed, it checks out the **master** branch. After initialization, you should see a **.git/** directory and a **.gitignore** file in the base directory for standalone server content.

Using a Remote Git Repository

To use a remote Git repository, start the server with the `--git-repo=REMOTE_REPO` argument. The value of the argument can be a URL or a remote alias that you have manually added to the local Git configuration.

You can also specify an optional branch or tag name in the remote repository by adding the `--git-branch=GIT_BRANCH_NAME` argument when you start the server. This argument should name an existing branch or tag name as it will *not* be created if it does not exist. If you use a tag name, you put the repository in a detached HEAD state, meaning future commits are not attached to any branches.

If your Git repository requires authentication, you must also add the `--git-auth=AUTH_FILE_URL` argument when you start the server. This argument should be the URL to an Elytron configuration file containing the credentials required to connect to the Git repository. The following is an example of an Elytron configuration file that can be used for authentication.

```
<?xml version="1.0" encoding="UTF-8"?>
```



```

<configuration>
  <authentication-client xmlns="urn:elytron:client:1.2">
    <authentication-rules>
      <rule use-configuration="test-login">
        </rule>
      </authentication-rules>
    <authentication-configurations>
      <configuration name="test-login">
        <sasl-mechanism-selector selector="BASIC" />
        <set-user-name name="eap-user" />
        <credentials>
          <clear-password password="my_api_key" />
        </credentials>
        <set-mechanism-realm name="testRealm" />
      </configuration>
    </authentication-configurations>
  </authentication-client>
</configuration>

```



NOTE

Although Git supports SSH authentication, Elytron does not; therefore, it is currently only possible to specify credentials to authenticate with HTTP or HTTPS, not SSH.

The following is an example of a command to start the server with the full profile, using the **1.0.x** branch of the remote **eap-configuration** repository, and passing the URL to an Elytron configuration file containing authentication credentials.

```

$ EAP_HOME/bin/standalone.sh --git-repo=https://github.com/MY_GIT_ID/eap-configuration.git --git-branch=1.0.x --git-auth=file:///home/USER_NAME/github-wildfly-config.xml --server-config=standalone-full.xml

```

If you start the server with the argument to use a remote Git repository, JBoss EAP checks whether the **jboss.server.base.dir** directory is already configured for Git. If not, JBoss EAP deletes the existing configuration files in the **jboss.server.base.dir** directory and replaces them with the remote Git configuration data. JBoss EAP checks out a branch name passed by the **--git-branch** argument. If that argument is not passed, it checks out the **master** branch. Once this process is complete, you should see a **.git/** directory and a **.gitignore** file in the base directory for standalone server content.



WARNING

If you later start the server passing a different **--git-repo** URL or **--git-branch** name than was originally used, you will see the error message **java.lang.RuntimeException: WFLYSRV0268: Failed to pull the repository GIT_REPO_NAME** when you attempt to start the server. This is because JBoss EAP attempts to pull configuration data from a different repository and branch than the one that is currently configured in the **jboss.server.base.dir** directory and the Git pull results in conflicts.

Publishing Remote Configuration Data When Using Git

You can push your Git repository changes to the remote repository using the management CLI **publish-configuration** operation. Because JBoss EAP pulls the configuration from the remote Git repository during the boot process when you start the server, this allows you to share the configuration data across multiple servers. You can only use this operation with a remote repository. It does not work for a local repository.

The following management CLI operation publishes the configuration data to the remote **eap-configuration** repository.

```
:publish-configuration(location="https://github.com/MY_GIT_ID/eap-configuration.git")
{"outcome" => "success"}
```

Using Snapshots with Git

In addition to using the Git commit history to track configuration changes, you can also take snapshots to preserve a configuration at a specific point in time. You can list the snapshots and delete them.

Taking Snapshots When Using Git

Snapshots are stored as tags in Git. You specify the snapshot tag name and commit message as arguments on the **take-snapshot** operation.

The following management CLI operation takes a snapshot and names the tag "snapshot-01".

```
:take-snapshot(name="snapshot-01", comment="1st snapshot")
{
  "outcome" => "success",
  "result" => "1st snapshot"
}
```

Listing Snapshots When Using Git

You can list all of the snapshot tags using the **list-snapshots** operation.

The following management CLI operation lists the snapshot tags.

```
:list-snapshots
{
  "outcome" => "success",
  "result" => {
    "directory" => "",
    "names" => [
      "snapshot : 1st snapshot",
      "refs/tags/snapshot-01",
      "snapshot2 : 2nd snapshot",
      "refs/tags/snapshot-02"
    ]
  }
}
```

Deleting Snapshots When Using Git

You can delete a specific snapshot by passing the tag name on the **delete-snapshot** operation.

The following management CLI operation deletes the snapshot with the tag name "snapshot-01".

```
:delete-snapshot(name="snapshot-01")
{"outcome" => "success"}
```

3.7. FILE SYSTEM PATHS

JBoss EAP uses logical names for file system paths. Other areas of the configuration can then reference the paths using their logical name, avoiding the need to use absolute paths for each instance and allowing specific host configurations to resolve to universal logical names.

For example, the default **logging** subsystem configuration declares **jboss.server.log.dir** as the logical name for the server log directory.

Example: Relative Path Example for the Server Log Directory

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

JBoss EAP automatically provides a number of standard paths without any need for the user to configure them in a configuration file.

Table 3.5. Standard Paths

Property	Description
java.home	The Java installation directory
jboss.controller.temp.dir	A common alias for standalone servers and managed domains. The directory to be used for temporary file storage. Equivalent to jboss.domain.temp.dir in a managed domain, and jboss.server.temp.dir on a standalone server.
jboss.domain.base.dir	The base directory for domain content.
jboss.domain.config.dir	The directory that contains the domain configuration.
jboss.domain.data.dir	The directory that the domain will use for persistent data file storage.
jboss.domain.log.dir	The directory that the domain will use for persistent log file storage.
jboss.domain.temp.dir	The directory that the domain will use for temporary file storage.
jboss.domain.deployment.dir	The directory that the domain will use for storing deployed content.
jboss.domain.servers.dir	The directory that the domain will use for storing outputs of the managed domain instances.
jboss.home.dir	The root directory of the JBoss EAP distribution.
jboss.server.base.dir	The base directory for standalone server content.
jboss.server.config.dir	The directory that contains the standalone server configuration.
jboss.server.data.dir	The directory the standalone server will use for persistent data file storage.

Property	Description
<code>jboss.server.log.dir</code>	The directory the standalone server will use for log file storage.
<code>jboss.server.temp.dir</code>	The directory the standalone server will use for temporary file storage.
<code>jboss.server.deploy.dir</code>	The directory that the standalone server will use for storing deployed content.
<code>user.dir</code>	The user's current working directory.
<code>user.home</code>	The user home directory.

You can [override a standard path](#) or [add a custom path](#).

3.7.1. View File System Paths

Use the following management CLI command to list the file system paths:

```
ls /path
```



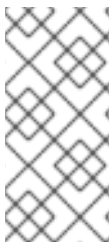
NOTE

In a managed domain, you can list the file system paths for a specific server using the following management CLI command:

```
ls /host=HOST_NAME/server=SERVER_NAME/path
```

Use the following management CLI command to read the value of a file system path:

```
/path=PATH_NAME:read-resource
```



NOTE

In a managed domain, you can read the value of a file system path for a specific server using the following management CLI command:

```
/host=HOST_NAME/server=SERVER_NAME/path=PATH_NAME:read-resource
```

3.7.2. Override a Standard Path

You can override the default locations of the standard paths that begin with **jboss.server.*** or **jboss.domain.***. This can be done in one of two ways:

- Pass in the command-line argument when you start the server. For example:

```
$ EAP_HOME/bin/standalone.sh -Djboss.server.log.dir=/var/log
```

- Modify the **JAVA_OPTS** variable in the server configuration file, either **standalone.conf** or **domain.conf**, to contain the new location. For example:

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.log.dir=/var/log"
```

Overriding a Managed Domain's Standard Paths

In this example, the objective is to store domain files in the **/opt/jboss_eap/domain_data** directory, and give each top-level directory a custom name. The default directory grouping, **by-server**, is used.

- Log files are to be stored in the **all_logs** subdirectory
- Data files are to be stored in the **all_data** subdirectory
- Temporary files are to be stored in the **all_temp** subdirectory
- Servers' files are to be stored in the **all_servers** subdirectory

To achieve this configuration, you would override several system properties when starting JBoss EAP.

```
$ EAP_HOME/bin/domain.sh -Djboss.domain.temp.dir=/opt/jboss_eap/domain_data/all_temp -
Djboss.domain.log.dir=/opt/jboss_eap/domain_data/all_logs -
Djboss.domain.data.dir=/opt/jboss_eap/domain_data/all_data -
Djboss.domain.servers.dir=/opt/jboss_eap/domain_data/all_servers
```

The resulting path structure will be as follows:

```
/opt/jboss_eap/domain_data/
├── all_data
├── all_logs
├── all_servers
│   ├── server-one
│   │   ├── data
│   │   ├── log
│   │   └── tmp
│   └── server-two
│       ├── data
│       ├── log
│       └── tmp
└── all_temp
```

3.7.3. Add a Custom Path

You can add a custom file system path using the management CLI or the management console.

- From the management CLI, you can add a new path using the following management CLI command.

```
/path=my.custom.path:add(path=/my/custom/path)
```

- From the management console, you can configure file system paths by navigating to the **Configuration** tab, selecting **Paths**, and clicking **View**. From there, you can add, modify, and remove paths.

You can then use this custom path in your configuration. For example, the below log handler uses a custom path for its relative path.

```
<subsystem xmlns="urn:jboss:domain:logging:6.0">
...
<periodic-rotating-file-handler name="FILE" autoflush="true">
  <formatter>
    <named-formatter name="PATTERN"/>
  </formatter>
  <file relative-to="my.custom.path" path="server.log"/>
  <suffix value=".yyyy-MM-dd"/>
  <append value="true"/>
</periodic-rotating-file-handler>
...
</subsystem>
```

3.8. DIRECTORY GROUPING

In a managed domain, each server's files are stored in the **EAP_HOME/domain** directory. You can specify how to organize the subdirectories for servers using the host controller's **directory-grouping** attribute. Directories can be grouped either by *server* or by *type*. By default, directories are grouped by *server*.

Directory Grouping by Server

By default, directories are grouped by server. If your administration is *server-centric*, this configuration is recommended. For example, it allows backups and log file handling to be configured per server instance.

If JBoss EAP is installed using the ZIP installation method, the default directory structure (grouped by server) will be as follows.

```
EAP_HOME/domain
├── servers
│   ├── server-one
│   │   ├── data
│   │   ├── tmp
│   │   └── log
│   └── server-two
│       ├── data
│       ├── tmp
│       └── log
```

To group domain directories by server, enter the following management CLI command:

```
/host=HOST_NAME:write-attribute(name=directory-grouping,value=by-server)
```

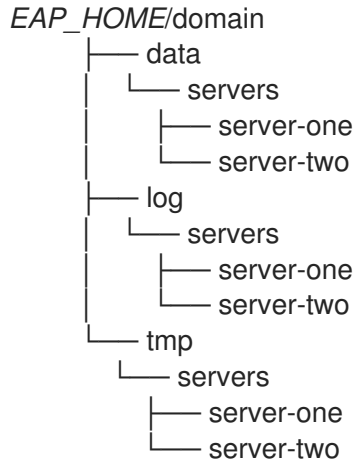
This will update the host controller's **host.xml** configuration file:

```
<servers directory-grouping="by-server">
  <server name="server-one" group="main-server-group"/>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```

Directory Grouping by Type

Instead of grouping directories by server, you can instead group by file type. If your administration is *file type-centric*, this configuration is recommended. For example, this would allow you to easily back up only **data** files.

If JBoss EAP is installed using the ZIP installation method and the domain's files are grouped by type, the directory structure will be as follows.



To group domain directories by type, enter the following management CLI command:

```
/host=HOST_NAME:write-attribute(name=directory-grouping,value=by-type)
```

This will update the host controller's **host.xml** configuration file:

```
<servers directory-grouping="by-type">
  <server name="server-one" group="main-server-group"/>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```

3.9. SYSTEM PROPERTIES

You can use Java system properties to configure many JBoss EAP options, as well as set any name-value pair for use within the application server.

System properties can be used to override default values in the JBoss EAP configuration. For example, the following XML configuration for the public interface bind address shows that it can be set by the **jboss.bind.address** system property, but if the system property is not provided, it will default to **127.0.0.1**.

```
<inet-address value="${jboss.bind.address:127.0.0.1}"/>
```

There are a few ways you can set system properties in JBoss EAP, including:

- passing the system property to the JBoss EAP startup script
- using the management CLI
- using the management console

- using the **JAVA_OPTS** environment variable

If you use a JBoss EAP managed domain, system properties can be applied to either the whole domain, a specific server group, a specific host and all its server instances, or just to one specific server instance. As with most other JBoss EAP domain settings, a system property set at a more specific level will override a more abstract one. See the [Domain Management](#) chapter for more information.

Passing a System Property to the Startup Script

You can pass a system property to the JBoss EAP startup script by using the **-D** argument. For example:

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=192.168.1.2
```

This method of setting the system property is especially useful for JBoss EAP options that need to be set before JBoss EAP starts.

Setting a System Property Using the Management CLI

Using the management CLI, you can set a system property using the following syntax:

```
/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

For example:

```
/system-property=jboss.bind.address:add(value=192.168.1.2)
```

When setting system properties using the management CLI, some JBoss EAP options, including the above example of **jboss.bind.address**, will only take effect after the next server restart.

For a managed domain, the above example configures a system property for the entire domain, but you can also set or override system properties at more specific levels of the domain configuration.

Setting a System Property Using the Management Console

- For a standalone JBoss EAP server, you can configure system properties in the management console under the **Configuration** tab. Select **System Properties**, and click the **View** button.
- For a managed domain:
 - Domain-level system properties can be set in the **Configuration** tab. Select **System Properties**, and click the **View** button.
 - Server group and server-level system properties can be set in the **Runtime** tab. Select the server group or server you want to configure, click the **View** button next to the server group or server name, and select the **System Properties** tab.
 - Host-level system properties can be set in the **Runtime** tab. Select the host you want to configure, then using the drop-down menu next to the host name, select **Properties**.

Setting a System Property Using JAVA_OPTS

System properties can also be configured using the **JAVA_OPTS** environment variable. There are many ways to modify **JAVA_OPTS**, but JBoss EAP provides a configuration file for setting **JAVA_OPTS** that is used by the JBoss EAP process.

For a standalone server, this file is **EAP_HOME/bin/standalone.conf**, or, for a managed domain, it is **EAP_HOME/bin/domain.conf**. For Microsoft Windows systems these files have a **.bat** extension.



NOTE

For an RPM installation, the [RPM service configuration file](#) is the preferred location to modify **JAVA_OPTS** to configure system properties. For more information, see [Configure RPM Service Properties](#).

Add your system property definition to **JAVA_OPTS** in the relevant configuration file. The examples below demonstrate setting the bind address on a Red Hat Enterprise Linux system.

- For **standalone.conf**, add your **JAVA_OPTS** system property definition at the end of the file. For example:

```
...
# Set the bind address
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address=192.168.1.2"
```

- For **domain.conf**, **JAVA_OPTS** must be set before the process controller **JAVA_OPTS** setting. For example:

```
...
# Set the bind address
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address=192.168.1.2"

# The ProcessController process uses its own set of java options
if [ "x$PROCESS_CONTROLLER_JAVA_OPTS" = "x" ]; then
...

```

3.10. MANAGEMENT AUDIT LOGGING

You can enable audit logging for the management interfaces, which will log all operations performed using the management console, management CLI, or custom application that uses the Management API. Audit log entries are stored in JSON format. By default, audit logging is disabled.

You can configure audit logging to output to a [file](#) or to a [syslog server](#).



NOTE

Login and logout events cannot be audited as there is no authenticated session in JBoss EAP. Instead, audit messages are logged when an operation is received from the user.

Standalone Server Audit Logging

Though disabled by default, the default audit logging configuration writes to a file.

```
<audit-log>
  <formatters>
    <json-formatter name="json-formatter"/>
  </formatters>
  <handlers>
    <file-handler name="file" formatter="json-formatter" path="audit-log.log" relative-
to="jboss.server.data.dir"/>
  </handlers>
  <logger log-boot="true" log-read-only="false" enabled="false">
    <handlers>
```

```

        <handler name="file"/>
    </handlers>
</logger>
</audit-log>

```

This configuration can be read using the following management CLI command.

```
/core-service=management/access=audit:read-resource(recursive=true)
```

See [Enable Audit Logging](#) to enable audit logging for a standalone server.

Managed Domain Audit Logging

Though disabled by default, the default audit logging configuration writes a file for each host and for each server.

```

<audit-log>
  <formatters>
    <json-formatter name="json-formatter"/>
  </formatters>
  <handlers>
    <file-handler name="host-file" formatter="json-formatter" relative-to="jboss.domain.data.dir"
path="audit-log.log"/>
    <file-handler name="server-file" formatter="json-formatter" relative-to="jboss.server.data.dir"
path="audit-log.log"/>
  </handlers>
  <logger log-boot="true" log-read-only="false" enabled="false">
    <handlers>
      <handler name="host-file"/>
    </handlers>
  </logger>
  <server-logger log-boot="true" log-read-only="false" enabled="false">
    <handlers>
      <handler name="server-file"/>
    </handlers>
  </server-logger>
</audit-log>

```

This configuration can be read using the following management CLI command.

```
/host=HOST_NAME/core-service=management/access=audit:read-resource(recursive=true)
```

See [Enable Audit Logging](#) to enable audit logging for a managed domain.

3.10.1. Enable Management Audit Logging

JBoss EAP is preconfigured with file handlers for audit logging, though audit logging is disabled by default. The management CLI command to enable audit logging depends on whether you are running as a standalone server or in a managed domain. See [Management Audit Logging Attributes](#) for file handler attributes.

The following instructions enable **NATIVE** and **HTTP** audit logging. To configure JMX audit logging, see [Enable JMX Management Audit Logging](#).

To set up syslog audit logging, see [Send Management Audit Logging to a Syslog Server](#).

Enable Standalone Server Audit Logging

Audit logging can be enabled using the following command.

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

By default, this will write the audit log to **EAP_HOME/standalone/data/audit-log.log**.

Enable Managed Domain Audit Logging

The default audit logging configuration for a managed domain is preconfigured to write an audit log for each host and each server.

Audit logging for each host can be enabled using the following command.

```
/host=HOST_NAME/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

By default, this will write the audit logs to **EAP_HOME/domain/data/audit-log.log**.

Audit logging for each server can be enabled using the following command.

```
/host=HOST_NAME/core-service=management/access=audit/server-logger=audit-log:write-attribute(name=enabled,value=true)
```

By default, this will write the audit logs to **EAP_HOME/domain/servers/SERVER_NAME/data/audit-log.log**.

3.10.2. Enable JMX Management Audit Logging

JBoss EAP is preconfigured with file handlers for JMX audit logging, though these logs are disabled by default. The management CLI command to enable audit logging depends on whether you are running as a standalone server or a managed domain.

To configure **NATIVE** or **HTTP** audit logging, see [Enable Management Audit Logging](#).

Enable Standalone Server JMX Audit Logging

JMX audit logging can be enabled for a standalone server using the following commands.

```
/subsystem=jmx/configuration=audit-log:add()  
/subsystem=jmx/configuration=audit-log/handler=file:add()
```

This enables JMX audit logging, and then uses the defined **file** handler to write these logs to **EAP_HOME/standalone/data/audit-log.log**.

Enable Managed Domain JMX Audit Logging

JMX audit logging can be enabled for each host and profile in a managed domain.

Enable JMX Audit Logging for a Host

1. Enable audit logging in the host's **jmx** subsystem.

```
/host=HOST_NAME/subsystem=jmx/configuration=audit-log:add()
```

2. Once audit logging for the **jmx** subsystem has been enabled, a handler can be defined for the host with the following command.

```
/host=HOST_NAME/subsystem=jmx/configuration=audit-log/handler=host-file:add()
```

By default, this will write the JMX audit logs to **EAP_HOME/domain/data/audit-log.log**.

Enable JMX Audit Logging for a Profile

1. Enable audit logging in the profile's **jmx** subsystem.

```
/profile=PROFILE_NAME/subsystem=jmx/configuration=audit-log:add()
```

2. Once audit logging for the **jmx** subsystem has been enabled, a handler can be defined for the profile with the following command.

```
/profile=PROFILE_NAME/subsystem=jmx/configuration=audit-log/handler=server-file:add()
```

By default, this will write the JMX audit logs to **EAP_HOME/domain/servers/SERVER_NAME/data/audit-log.log**.

3.10.3. Send Management Audit Logging to a Syslog Server

A syslog handler specifies the parameters by which audit log entries are sent to a syslog server, specifically the syslog server's host name and the port on which the syslog server is listening. Sending audit logging to a syslog server provides more security options than logging to a local file or local syslog server. Multiple syslog handlers can be defined and be active at the same time.

By default, audit logging is preconfigured to output to a file when enabled. Use the following steps to set up and enable audit logging to a syslog server. See [Management Audit Logging Attributes](#) for syslog handler attributes.

1. Add a syslog handler.

Create the syslog handler, specifying the host and port of the syslog server. In a managed domain, you must precede the **/core-service** commands with **/host=HOST_NAME**.

```
batch
/core-service=management/access=audit/syslog-
handler=SYSLOG_HANDLER_NAME:add(formatter=json-formatter)
/core-service=management/access=audit/syslog-
handler=SYSLOG_HANDLER_NAME/protocol=udp:add(host=HOST_NAME,port=PORT)
run-batch
```

NOTE

The parameters to pass in differ depending on the protocol specified.

To configure the handler to use TLS to communicate securely with the syslog server, you must also configure the authentication, for example:

```
/core-service=management/access=audit/syslog-
handler=SYSLOG_HANDLER_NAME/protocol=tls/authentication=truststore:ad
d(keystore-path=PATH_TO_TRUSTSTORE,keystore-
password=TRUSTSTORE_PASSWORD)
```

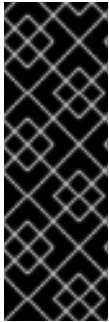
2. Add a reference to the syslog handler.

In a managed domain, you must precede this command with **/host=HOST_NAME**.

```
/core-service=management/access=audit/logger=audit-  
log/handler=SYSLOG_HANDLER_NAME:add
```

3. Enable audit logging.

See [Enable Management Audit Logging](#) to enable audit logging.



IMPORTANT

Enabling audit logging to a syslog server in JBoss EAP will not work unless logging is enabled in the operating system as well.

For more information on **rsyslog** configurations on Red Hat Enterprise Linux, see the *Basic Configuration of Rsyslog* section of the *System Administrator's Guide* for Red Hat Enterprise Linux at <https://access.redhat.com/documentation/en/red-hat-enterprise-linux/>.

3.10.4. Read Audit Log Entries

Audit log entries output to files are best viewed with a text *viewer*, while those output to a syslog server are best viewed using a syslog viewer application.



NOTE

Using a text *editor* for viewing log files is not recommended as some may prevent further log entries being written to the log file.

The audit log entries are stored in JSON format. Each log entry begins with an optional timestamp, followed by the fields in the below table.

Table 3.6. Management Audit Log Fields

Field Name	Description
access	<p>This can have one of the following values:</p> <ul style="list-style-type: none"> ● NATIVE - The operation came in through the native management interface. ● HTTP - The operation came in through the domain HTTP interface. ● JMX - The operation came in through the jmx subsystem.
booting	<p>Has the value true if the operation was executed during the bootup process, or false if it was executed once the server is up and running.</p>
domainUUID	<p>An ID to link together all operations as they are propagated from the domain controller to its servers, slave host controllers, and slave host controller servers.</p>

Field Name	Description
ops	The operations being executed. This is a list of the operations serialized to JSON. At boot, this is the operations resulting from parsing the XML. Once booted, the list typically contains a single entry.
r/o	Has the value true if the operation does not change the management model, or false if it does.
remote-address	The address of the client executing this operation.
success	Has the value true if the operation was successful, or false if it was rolled back.
type	This can have the value core , meaning it is a management operation, or jmx , meaning it comes from the jmx subsystem.
user	The username of the authenticated user. If the operation occurred using the management CLI on the same machine as the running server, the special user \$local is used.
version	The version number of the JBoss EAP instance.

3.11. SERVER LIFECYCLE EVENT NOTIFICATIONS

You can set up notifications for server lifecycle events using the JBoss EAP [core-management subsystem](#) or [JMX](#). A change in server runtime configuration state or server running state will trigger a notification.

The server runtime configuration states for JBoss EAP are **STARTING**, **RUNNING**, **RELOAD_REQUIRED**, **RESTART_REQUIRED**, **STOPPING**, and **STOPPED**.

The server running states for JBoss EAP are **STARTING**, **NORMAL**, **ADMIN_ONLY**, **PRE_SUSPEND**, **SUSPENDING**, **SUSPENDED**, **STOPPING**, and **STOPPED**.

3.11.1. Monitor Server Lifecycle Events Using the Core Management Subsystem

You can register a listener to the JBoss EAP **core-management** subsystem to monitor for server lifecycle events. The following steps show how to create and register an example listener that logs the events to a file.

1. Create the listener.
Create an implementation of **org.wildfly.extension.core.management.client.ProcessStateListener**, like the example below.

Example: Listener Class

```
package org.simple.lifecycle.events.listener;
```

```

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

import org.wildfly.extension.core.management.client.ProcessStateListener;
import org.wildfly.extension.core.management.client.ProcessStateListenerInitParameters;
import org.wildfly.extension.core.management.client.RunningStateChangeEvent;
import org.wildfly.extension.core.management.client.RuntimeConfigurationStateChangeEvent;

public class SimpleListener implements ProcessStateListener {

    private File file;
    private FileWriter fileWriter;
    private ProcessStateListenerInitParameters parameters;

    public void init(ProcessStateListenerInitParameters parameters) {
        this.parameters = parameters;
        this.file = new File(parameters.getInitProperties().get("file"));
        try {
            fileWriter = new FileWriter(file, true);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void cleanup() {
        try {
            fileWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            fileWriter = null;
        }
    }

    public void runtimeConfigurationStateChanged(RuntimeConfigurationStateChangeEvent
    evt) {
        try {
            fileWriter.write(String.format("Runtime configuration state change for %s: %s to
%s\n", parameters.getProcessType(), evt.getOldState(), evt.getNewState()));
            fileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void runningStateChanged(RunningStateChangeEvent evt) {
        try {
            fileWriter.write(String.format("Running state change for %s: %s to %s\n",
parameters.getProcessType(), evt.getOldState(), evt.getNewState()));
            fileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```
}
}
}
```



NOTE

Keep the following in mind when implementing the listener.

- In the event of a server reload, listeners stop listening while the server attempts to stop, and the listeners are reloaded when the server starts. Because of this, implementations must ensure that they can be loaded, initialized, and removed properly several times inside the same JVM.
- Notifications to the listeners are blocking to allow reactions to server state changes. Implementations must ensure that they do not block or deadlock.
- Each listener instance is executed in its own thread and the order is not guaranteed.

2. Compile the class and package it into a JAR.

Note that to compile, you need to depend on the **org.wildfly.core:wildfly-core-management-client** Maven module.

3. Add the JAR as a JBoss EAP module.

Use the following management CLI command and provide the module name and path to the JAR.

```
module add --name=org.simple.lifecycle.events.listener --
dependencies=org.wildfly.extension.core-management-client --resources=/path/to/simple-
listener-0.0.1-SNAPSHOT.jar
```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

4. Register the listener.

Add the listener to the **core-management** subsystem using the following management CLI command. Specify the class, module, and file location to log the server lifecycle events.

```
/subsystem=core-management/process-state-listener=my-simple-
listener:add(class=org.simple.lifecycle.events.listener.SimpleListener,
module=org.simple.lifecycle.events.listener,properties={file=/path/to/my-listener-output.txt})
```


Now, server lifecycle events will be logged to the **my-listener-output.txt** file based on the **SimpleListener** class above. For example, issuing a **:suspend** command in the management CLI will output the following to the **my-listener-output.txt** file.

```
Running state change for STANDALONE_SERVER: normal to suspending
Running state change for STANDALONE_SERVER: suspending to suspended
```

This shows that the running state changed from **normal** to **suspending**, and then from **suspending** to **suspended**.

3.11.2. Monitor Server Lifecycle Events Using JMX Notifications

You can register a JMX notification listener to monitor for server lifecycle events. The following steps show how to create and add an example listener that logs events to a file.

1. Create the listener.
Create an implementation of **javax.management.NotificationListener**, like the example below.

Example: Listener Class

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

import javax.management.AttributeChangeNotification;
import javax.management.Notification;
import javax.management.NotificationListener;

import org.jboss.logging.Logger;

public class StateNotificationListener implements NotificationListener {

    public static final String RUNTIME_CONFIGURATION_FILENAME = "runtime-configuration-notifications.txt";
    public static final String RUNNING_FILENAME = "running-notifications.txt";
    private final Path targetFile;

    public StateNotificationListener() {
        this.targetFile = Paths.get("notifications/data").toAbsolutePath();
        init(targetFile);
    }

    protected Path getRuntimeConfigurationTargetFile() {
        return this.targetFile.resolve(RUNTIME_CONFIGURATION_FILENAME);
    }

    protected Path getRunningConfigurationTargetFile() {
        return this.targetFile.resolve(RUNNING_FILENAME);
    }
}
```

```

protected final void init(Path targetFile) {
    try {
        Files.createDirectories(targetFile);

        if (!Files.exists(targetFile.resolve(RUNTIME_CONFIGURATION_FILENAME))) {
            Files.createFile(targetFile.resolve(RUNTIME_CONFIGURATION_FILENAME));
        }

        if (!Files.exists(targetFile.resolve(RUNNING_FILENAME))) {
            Files.createFile(targetFile.resolve(RUNNING_FILENAME));
        }
    } catch (IOException ex) {
        Logger.getLogger(StateNotificationListener.class).error("Problem handling JMX
Notification", ex);
    }
}

@Override
public void handleNotification(Notification notification, Object handback) {
    AttributeChangeNotification attributeChangeNotification = (AttributeChangeNotification)
notification;
    if ("RuntimeConfigurationState".equals(attributeChangeNotification.getAttributeName())) {
        writeNotification(attributeChangeNotification, getRuntimeConfigurationTargetFile());
    } else {
        writeNotification(attributeChangeNotification, getRunningConfigurationTargetFile());
    }
}

private void writeNotification(AttributeChangeNotification notification, Path path) {
    try (BufferedWriter in = Files.newBufferedWriter(path, StandardCharsets.UTF_8,
StandardOpenOption.APPEND)) {
        in.write(String.format("%s %s %s %s", notification.getType(),
notification.getSequenceNumber(), notification.getSource().toString(),
notification.getMessage()));
        in.newLine();
        in.flush();
    } catch (IOException ex) {
        Logger.getLogger(StateNotificationListener.class).error("Problem handling JMX
Notification", ex);
    }
}
}

```

2. Register the notification listener.
Add the notification listener to the **MBeanServer**.

Example: Add a Notification Listener

```

MBeanServer server = ManagementFactory.getPlatformMBeanServer();
server.addNotificationListener(ObjectName.getInstance("jboss.root:type=state"), new
StateNotificationListener(), null, null);

```

3. Package and deploy to JBoss EAP.

Server lifecycle events are now logged to a file based on the **StateNotificationListener** class above. For example, issuing a **:suspend** command in the management CLI outputs the following to the **running-notifications.txt** file.

```
jmx.attribute.change 5 jboss.root:type=state The attribute 'RunningState' has changed from 'normal'
to 'suspending'
jmx.attribute.change 6 jboss.root:type=state The attribute 'RunningState' has changed from
'suspending' to 'suspended'
```

This shows that the running state changed from **normal** to **suspending**, and then from **suspending** to **suspended**.

CHAPTER 4. NETWORK AND PORT CONFIGURATION

4.1. INTERFACES

JBoss EAP references named interfaces throughout the configuration. This allows the configuration to reference individual interface declarations with logical names, rather than requiring the full details of the interface at each use.

This also allows for easier configuration in a managed domain, where network interface details can vary across multiple machines. Each server instance can correspond to a logical name group.

The **standalone.xml**, **domain.xml**, and **host.xml** files all include interface declarations. There are several preconfigured interface names, depending on which default configuration is used. The **management** interface can be used for all components and services that require the management layer, including the HTTP management endpoint. The **public** interface can be used for all application-related network communications. The **unsecure** interface is used for IIOP sockets in the standard configuration. The **private** interface is used for JGroups sockets in the standard configuration.

4.1.1. Default Interface Configurations

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="private">
    <inet-address value="${jboss.bind.address.private:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

By default, JBoss EAP binds these interfaces to **127.0.0.1**, but these values can be overridden at runtime by setting the appropriate property. For example, the **inet-address** of the **public** interface can be set when starting JBoss EAP as a standalone server with the following command.

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

Alternatively, you can use the **-b** switch on the server start command line. For more information about server start options, see [Server Runtime Arguments](#).



IMPORTANT

If you modify the default network interfaces or ports that JBoss EAP uses, you must also remember to change any scripts that use the modified interfaces or ports. These include JBoss EAP service scripts, as well as remembering to specify the correct interface and port when accessing the management console or management CLI.

4.1.2. Configuring Interfaces

Network interfaces are declared by specifying a logical name and selection criteria for the physical interface. The selection criteria can reference a wildcard address or specify a set of one or more characteristics that an interface or address must have in order to be a valid match. For a listing of all available interface selection criteria, see the [Interface Attributes](#) section.

Interfaces can be configured using the management console or the management CLI. Below are several examples of adding and updating interfaces. The management CLI command is shown first, followed by the corresponding configuration XML.

Add an Interface with a NIC Value

Add a new interface with a NIC value of **eth0**.

```
/interface=external:add(nic=eth0)
```

```
<interface name="external">
  <nic name="eth0"/>
</interface>
```

Add an Interface with Several Conditional Values

Add a new interface that matches any interface/address on the correct subnet if it is up, supports multicast, and is not point-to-point.

```
/interface=default:add(subnet-match=192.168.0.0/16,up=true,multicast=true,not={point-to-point=true})
```

```
<interface name="default">
  <subnet-match value="192.168.0.0/16"/>
  <up/>
  <multicast/>
  <not>
    <point-to-point/>
  </not>
</interface>
```

Update an Interface Attribute

Update the **public** interface's default **inet-address** value, keeping the **jboss.bind.address** property to allow for this value to be set at runtime.

```
/interface=public:write-attribute(name=inet-address,value="${jboss.bind.address:192.168.0.0}")
```

```
<interface name="public">
  <inet-address value="${jboss.bind.address:192.168.0.0}"/>
</interface>
```

Add an Interface to a Server in a Managed Domain

```
/host=HOST_NAME/server-config=SERVER_NAME/interface=INTERFACE_NAME:add(inet-address=127.0.0.1)
```

```
<servers>
  <server name="SERVER_NAME" group="main-server-group">
    <interfaces>
      <interface name="INTERFACE_NAME">
        <inet-address value="127.0.0.1"/>
      </interface>
    </interfaces>
  </server>
</servers>
```

```
</interface>
</interfaces>
</server>
</servers>
```

4.2. SOCKET BINDINGS

Socket bindings and socket binding groups allow you to define network ports and their relationship to the networking interfaces required for your JBoss EAP configuration. A socket binding is a named configuration for a socket. A socket binding group is a collection of socket binding declarations that are grouped under a logical name.

This allows other sections of the configuration to reference socket bindings by their logical name, rather than requiring the full details of the socket configuration at each use.

The declarations for these named configurations can be found in the **standalone.xml** and **domain.xml** configuration files. A standalone server contains only one socket binding group, while a managed domain can contain multiple groups. You can create a socket binding group for each server group in the managed domain, or share a socket binding group between multiple server groups.

The ports JBoss EAP uses by default depend on which socket binding groups are used and the requirements of your individual deployments.

There are three types of socket bindings that can be defined in a socket binding group in the JBoss EAP configuration:

Inbound Socket Bindings

The **socket-binding** element is used to configure inbound socket bindings for the JBoss EAP server. The default JBoss EAP configurations provide several preconfigured **socket-binding** elements, for example, for HTTP and HTTPS traffic. Another example can be found in the [Broadcast Groups](#) section of *Configuring Messaging for JBoss EAP*.

Attributes for this element can be found in the [Inbound Socket Binding Attributes](#) table.

Remote Outbound Socket Bindings

The **remote-destination-outbound-socket-binding** element is used to configure outbound socket bindings for destinations that are remote to the JBoss EAP server. The default JBoss EAP configurations provide an example remote destination socket binding that can be used for a mail server. Another example can be found in the [Using the Integrated Artemis Resource Adapter for Remote Connections](#) section of *Configuring Messaging for JBoss EAP*.

Attributes for this element can be found in the [Remote Outbound Socket Binding Attributes](#) table.

Local Outbound Socket Bindings

The **local-destination-outbound-socket-binding** element is used to configure outbound socket bindings for destinations that are local to the JBoss EAP server. This type of socket binding is not expected to be commonly used.

Attributes for this element can be found in the [Local Outbound Socket Binding Attributes](#) table.

4.2.1. Management Ports

Management ports were consolidated in JBoss EAP 7. By default, JBoss EAP 7 uses port **9990** for both native management, used by the management CLI, and HTTP management, used by the web-based management console. Port **9999**, which was used as the native management port in JBoss EAP 6, is no

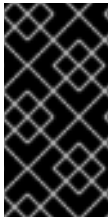
longer used but can still be enabled if desired.

If HTTPS is enabled for the management console, then port **9993** is used by default.

4.2.2. Default Socket Bindings

JBoss EAP ships with a socket binding group for each of the five predefined profiles (*default*, *ha*, *full*, *full-ha*, *load-balancer*).

For detailed information about the default socket bindings, such as default ports and descriptions, see the [Default Socket Bindings](#) section.



IMPORTANT

If you modify the default network interfaces or ports that JBoss EAP uses, you must also remember to change any scripts that use the modified interfaces or ports. These include JBoss EAP service scripts, as well as remembering to specify the correct interface and port when accessing the management console or management CLI.

Standalone Server

When running as a standalone server, only one socket binding group is defined per configuration file. Each standalone configuration file (**standalone.xml**, **standalone-ha.xml**, **standalone-full.xml**, **standalone-full-ha.xml**, **standalone-load-balancer.xml**) defines socket bindings for the technologies used by its corresponding profile.

For example, the default standalone configuration file (**standalone.xml**) specifies the below socket bindings.

```
<socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}" />
  <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9993}" />
  <socket-binding name="ajp" port="${jboss.ajp.port:8009}" />
  <socket-binding name="http" port="${jboss.http.port:8080}" />
  <socket-binding name="https" port="${jboss.https.port:8443}" />
  <socket-binding name="txn-recovery-environment" port="4712" />
  <socket-binding name="txn-status-manager" port="4713" />
  <outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25" />
  </outbound-socket-binding>
</socket-binding-group>
```

Managed Domain

When running in a managed domain, all socket binding groups are defined in the **domain.xml** file. There are five predefined socket binding groups:

- **standard-sockets**
- **ha-sockets**
- **full-sockets**
- **full-ha-sockets**

- **load-balancer-sockets**

Each socket binding group specifies socket bindings for the technologies used by its corresponding profile. For example, the **full-ha-sockets** socket binding group defines several **jgroups** socket bindings, which are used by the *full-ha* profile for high availability.

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-interface="public">
    <!-- Needed for server groups using the 'default' profile -->
    <socket-binding name="ajp" port="{jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="{jboss.http.port:8080}"/>
    <socket-binding name="https" port="{jboss.https.port:8443}"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'ha' profile -->
    ...
  </socket-binding-group>
  <socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full' profile -->
    ...
  </socket-binding-group>
  <socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile -->
    <socket-binding name="ajp" port="{jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="{jboss.http.port:8080}"/>
    <socket-binding name="https" port="{jboss.https.port:8443}"/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="jgroups-mping" interface="private" port="0" multicast-
address="{jboss.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" interface="private" port="7600"/>
    <socket-binding name="jgroups-udp" interface="private" port="55200" multicast-
address="{jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
    <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105" multicast-
port="23364"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="load-balancer-sockets" default-interface="public">
    <!-- Needed for server groups using the 'load-balancer' profile -->
    ...
  </socket-binding-group>
</socket-binding-groups>
```




NOTE

The socket configuration for the management interfaces is defined in the domain controller's **host.xml** file.

4.2.3. Configuring Socket Bindings

When defining a socket binding, you can configure the **port** and **interface** attributes, as well as multicast settings such as **multicast-address** and **multicast-port**. For details on all available socket bindings attributes, see the [Socket Binding Attributes](#) section.

Socket bindings can be configured using the management console or the management CLI. The following steps go through adding a socket binding group, adding a socket binding, and configuring socket binding settings using the management CLI.

1. Add a new socket binding group. Note that this step cannot be performed when running as a standalone server.

```
/socket-binding-group=new-sockets:add(default-interface=public)
```

2. Add a socket binding.

```
/socket-binding-group=new-sockets/socket-binding=new-socket-binding:add(port=1234)
```

3. Change the socket binding to use an interface other than the default, which is set by the socket binding group.

```
/socket-binding-group=new-sockets/socket-binding=new-socket-binding:write-attribute(name=interface,value=unsecure)
```

The following example shows how the XML configuration may look after the above steps have been completed.

```
<socket-binding-groups>
...
<socket-binding-group name="new-sockets" default-interface="public">
  <socket-binding name="new-socket-binding" interface="unsecure" port="1234"/>
</socket-binding-group>
</socket-binding-groups>
```

4.2.4. Viewing Socket Bindings and Open Ports for a Server

You can view the socket binding name and the open ports for a server from the management console. The information is visible when the server is in the following states:

- **running**
- **reload-required**
- **restart-required**

To view the socket bindings and the open ports for a server:

1. Access the management console and navigate to **Runtime**.

2. Click the server to view the socket binding name and the open ports in the right pane.

4.2.5. Port Offsets

A port offset is a numeric offset value added to all port values specified in the socket binding group for that server. This allows the server to inherit the port values defined in its socket binding group, with an offset to ensure that it does not conflict with any other servers on the same host. For instance, if the HTTP port of the socket binding group is **8080**, and a server uses a port offset of **100**, then its HTTP port is **8180**.

Below is an example of setting a port offset of **250** for a server in a managed domain using the management CLI.

```
/host=master/server-config=server-two/:write-attribute(name=socket-binding-port-offset,value=250)
```

Port offsets can be used for servers in a managed domain and for running multiple standalone servers on the same host.

You can pass in a port offset when starting a standalone server using the **jboss.socket.binding.port-offset** property.

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

4.3. IPV6 ADDRESSES

By default, JBoss EAP is configured to run using IPv4 addresses. The steps below show how to configure JBoss EAP to run using IPv6 addresses.

Configure the JVM Stack for IPv6 Addresses

Update the startup configuration to prefer IPv6 addresses.

1. Open the startup configuration file.
 - When running as a standalone server, edit the **EAP_HOME/bin/standalone.conf** file (or **standalone.conf.bat** for Windows Server).
 - When running in a managed domain, edit the **EAP_HOME/bin/domain.conf** file (or **domain.conf.bat** for Windows Server).
2. Set the **java.net.preferIPv4Stack** property to **false**.

```
-Djava.net.preferIPv4Stack=false
```

3. Append the **java.net.preferIPv6Addresses** property and set it to **true**.

```
-Djava.net.preferIPv6Addresses=true
```

The following example shows how the JVM options in the startup configuration file may look after making the above changes.

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms1303m -Xmx1303m -Djava.net.preferIPv4Stack=false"
```

```

    JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBoss_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
    JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv6Addresses=true"
else

```

Update Interface Declarations for IPv6 Addresses

The default interface values in the configuration can be changed to IPv6 addresses. For example, the below management CLI command sets the **management** interface to the IPv6 loopback address (**::1**).

```

/interface=management:write-attribute(name=inet-
address,value="{jboss.bind.address.management::1}")

```

The following example shows how the XML configuration may look after running the above command.

```

<interfaces>
  <interface name="management">
    <inet-address value="{jboss.bind.address.management::1}"/>
  </interface>
  ....
</interfaces>

```

CHAPTER 5. JBOSS EAP SECURITY

JBoss EAP offers the ability to configure security for its own interfaces and services as well as provide security for applications that are running on it.

- See the [Security Architecture](#) guide for an overview of general security concepts as well as JBoss EAP-specific security concepts.
- See [How to Configure Server Security](#) for information on securing JBoss EAP itself.
- See [How to Configure Identity Management](#) for information on providing security for applications deployed to JBoss EAP.
- See [How to Set Up SSO with Kerberos](#) for information on configuring single sign-on for JBoss EAP using Kerberos.
- See [How To Set Up SSO with SAML v2](#) for information on configuring single sign-on for JBoss EAP using SAML v2.

CHAPTER 6. JBOSS EAP CLASS LOADING

JBoss EAP uses a modular class loading system for controlling the class paths of deployed applications. This system provides more flexibility and control than the traditional system of hierarchical class loaders. Developers have fine-grained control of the classes available to their applications, and can configure a deployment to ignore classes provided by the application server in favor of their own.

The modular class loader separates all Java classes into logical groups called modules. Each module can define dependencies on other modules in order to have the classes from that module added to its own class path. Because each deployed JAR and WAR file is treated as a module, developers can control the contents of their application's class path by adding module configuration to their application.

6.1. MODULES

A module is a logical grouping of classes used for class loading and dependency management. JBoss EAP identifies two different types of modules: *static* and *dynamic*. The main difference between the two is how they are packaged.

Static Modules

Static modules are defined in the **EAP_HOME/modules/** directory of the application server. Each module exists as a subdirectory, for example, **EAP_HOME/modules/com/mysql/**. Each module directory then contains a slot subdirectory, which defaults to **main** and contains the **module.xml** configuration file and any required JAR files. All the application server-provided APIs are provided as static modules, including the Jakarta EE APIs as well as other APIs.

Example: MySQL JDBC Driver module.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

The module name, **com.mysql**, must match the directory structure for the module, excluding the slot name, **main**.

Creating custom static modules can be useful if many applications are deployed on the same server that use the same third-party libraries. Instead of bundling those libraries with each application, a module containing these libraries can be created and installed by an administrator. The applications can then declare an explicit dependency on the custom static modules.

The modules provided in JBoss EAP distributions are located in the **system** directory within the **EAP_HOME/modules** directory. This keeps them separate from any modules provided by third parties. Any Red Hat provided products that layer on top of JBoss EAP also install their modules within the **system** directory.

Users must ensure that custom modules are installed into the **EAP_HOME/modules** directory, using one directory per module. This ensures that custom versions of modules that already exist in the **system** directory are loaded instead of the shipped versions. In this way, user-provided modules will take precedence over system modules.

If you use the **JBOSS_MODULEPATH** environment variable to change the locations in which JBoss EAP searches for modules, then the product will look for a **system** subdirectory structure within one of the locations specified. A **system** structure must exist somewhere in the locations specified with **JBOSS_MODULEPATH**.



NOTE

Starting with JBoss EAP 7.1, the use of absolute paths in the **resource-root path** element of the **module.xml** file is also supported. This way, your resource libraries can be made accessible without needing to move them to the **EAP_HOME/modules** directory.

Example: Absolute Path in the module.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="oracle.jdbc">
  <resources>
    <resource-root path="/home/redhat/test.jar"/>
  </resources>
</module>
```

Dynamic Modules

Dynamic modules are created and loaded by the application server for each JAR or WAR deployment, or for each subdeployment in an EAR. The name of a dynamic module is derived from the name of the deployed archive. Because deployments are loaded as modules, they can configure dependencies and be used as dependencies by other deployments.

Modules are only loaded when required. This usually only occurs when an application is deployed that has explicit or implicit dependencies.

Predefined Modules

Starting with JBoss EAP 7.2, a set of predefined modules is available to you when you use the default module loader. The special module, **org.jboss.modules**, which includes all of the JBoss Modules API, is always available and is provided by JBoss Modules. The standard Java Platform Module System (JPMS) modules, which are provided in Java 9 and later, are also available by their standard names. When using JDK 8, the JDK 9 modules are emulated by JBoss Modules.

For a list of platform modules available on Java 9 or later, see the appropriate JDK documentation.

For the list the platform modules provided for Java 8, see [Platform Modules Provided for Java 8](#).

6.2. MODULE DEPENDENCIES

A module dependency is a declaration that one module requires the classes of one or more other modules in order to function. When JBoss EAP loads a module, the modular class loader parses the dependencies of that module and adds the classes from each dependency to its class path. If a specified dependency cannot be found, the module will fail to load.



NOTE

See the [Modules](#) section for complete details about modules and the modular class loading system.

Deployed applications, such as a JAR or WAR, are loaded as dynamic modules and make use of dependencies to access the APIs provided by JBoss EAP.

There are two types of dependencies: *explicit* and *implicit*.

Explicit Dependencies

Explicit dependencies are declared by the developer in a configuration file. A static module can declare dependencies in its **module.xml** file. A dynamic module can declare dependencies in the deployment's **MANIFEST.MF** or **jboss-deployment-structure.xml** deployment descriptor.

Implicit Dependencies

Implicit dependencies are added automatically by JBoss EAP when certain conditions or meta-data are found in a deployment. The Jakarta EE APIs supplied with JBoss EAP are examples of modules that are added by detection of implicit dependencies in deployments.

Deployments can also be configured to exclude specific implicit dependencies by using the **jboss-deployment-structure.xml** deployment descriptor file. This can be useful when an application bundles a specific version of a library that JBoss EAP will attempt to add as an implicit dependency.

Optional Dependencies

Explicit dependencies can be specified as optional. Failure to load an optional dependency will not cause a module to fail to load. However, if the dependency becomes available later it will *not* be added to the module's class path. Dependencies must be available when the module is loaded.

Export a Dependency

A module's class path contains only its own classes and that of its immediate dependencies. A module is not able to access the classes of the dependencies of one of its dependencies. However, a module can specify that an explicit dependency is exported. An exported dependency is provided to any module that depends on the module that exports it.

For example, Module *A* depends on Module *B*, and Module *B* depends on Module *C*. Module *A* can access the classes of Module *B*, and Module *B* can access the classes of Module *C*. Module *A* cannot access the classes of Module *C* unless:

- Module *A* declares an explicit dependency on Module *C*, or
- Module *B* exports its dependency on Module *C*.

Global Modules

A global module is a module that JBoss EAP provides as a dependency to every application. Any module can be made global by adding it to JBoss EAP's list of global modules. It does not require changes to the module.

See the [Define Global Modules](#) section for details.

6.3. CREATE A CUSTOM MODULE

Custom static modules can be added to make resources available for deployments running on JBoss EAP. You can create the module [manually](#) or by [using the management CLI](#).

Once you create the module, you must [add the module as a dependency](#) if its resources need to be made available to applications.

Create a Custom Module Manually

You can create a custom module manually using the following steps.

1. Create the appropriate directory structure in the **EAP_HOME/modules/** directory.

Example: Create MySQL JDBC Driver Directory Structure

```
$ cd EAP_HOME/modules/
$ mkdir -p com/mysql/main
```

2. Copy the JAR files or other necessary resources to the **main/** subdirectory.

Example: Copy MySQL JDBC Driver JAR

```
$ cp /path/to/mysql-connector-java-8.0.12.jar EAP_HOME/modules/com/mysql/main/
```

3. Create a **module.xml** file in the **main/** subdirectory, specifying the appropriate resources and dependencies in the file.

Example: MySQL JDBC Driver module.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Create a Custom Module Using the Management CLI

You can create a custom module using the **module add** management CLI command.

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

1. Start the JBoss EAP server.
2. Launch the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh
```


3. Use the **module add** management CLI command to add the new core module.

```
module add --name=MODULE_NAME --resources=PATH_TO_RESOURCE --
dependencies=DEPENDENCIES
```

Example: Create a MySQL Module

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api
```

See [Module Command Arguments](#) for the available arguments to customize this command, such as providing your own **module.xml** file, using an external module directory, or specifying an alternative slot for the module. You can also execute **module --help** for more details on using this command to add and remove modules.

Add the Module as a Dependency

In order for your application to be able to access this module's resources, you must add the module as a dependency.

- See the [Add an Explicit Module Dependency to a Deployment](#) section of the JBoss EAP *Development Guide* for adding application-specific dependencies using deployment descriptors.
- See the [Define Global Modules](#) section for instructions on adding modules as dependencies to all applications.

As an example, the following steps add a JAR file containing several properties files as a module and define a global module, so that an application can then load these properties.

1. Add the JAR file as a core module.

```
module add --name=myprops --resources=/path/to/properties.jar
```

2. Define this module as a global module so that it is made available to all deployments.

```
/subsystem=ee:list-add(name=global-modules,value={name=myprops})
```

3. The application can then retrieve the properties from one of the properties files contained within the JAR.

```
Thread.currentThread().getContextClassLoader().getResource("my.properties");
```

6.4. REMOVE A CUSTOM MODULE

Custom static modules can be removed [manually](#) or by [using the management CLI](#).

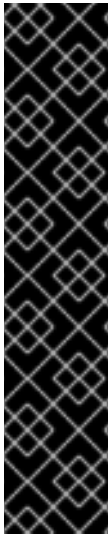
Remove a Custom Module Manually

Before manually removing a module, ensure that it is not required by deployed applications or elsewhere in the server configuration, such as by a datasource.

To remove a custom module, remove the module's directory under **EAP_HOME/modules/**, which includes its **module.xml** file and associated JAR files or other resources. For example, remove the **EAP_HOME/modules/com/mysql/main/** directory to remove a custom MySQL JDBC driver module in the **main** slot.

Remove a Custom Module Using the Management CLI

You can remove a custom module using the **module remove** management CLI command.



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

1. Start the JBoss EAP server.
2. Launch the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh
```

3. Use the **module remove** management CLI command to remove the custom module.

```
module remove --name=MODULE_NAME
```

- Use the **--slot** argument if the module to remove is in a slot other than **main**.

Example: Remove a MySQL Module

```
module remove --name=com.mysql
```

Execute **module --help** for more details on using this command to add and remove modules.

6.5. DEFINE GLOBAL MODULES

A list of global modules can be defined for JBoss EAP, which will add the modules as dependencies to all deployments.



NOTE

You must know the name of the modules that are to be configured as global modules. For the complete listing of the included modules and whether they are supported, see [Red Hat JBoss Enterprise Application Platform 7 Included Modules](#) on the Red Hat Customer Portal. See the [Dynamic Module Naming](#) section for naming conventions for modules in deployments.

Use the following management CLI command to define the list of global modules.

```
/subsystem=ee:write-attribute(name=global-modules,value=[{name=MODULE_NAME_1},
{name=MODULE_NAME_2}]
```

Use the following management CLI command to add a single module to the list of existing global modules.

```
/subsystem=ee:list-add(name=global-modules,value={name=MODULE_NAME})
```

Global modules can also be added and removed using the management console by navigating to the **EE** subsystem from the **Configuration** tab and selecting the **Global Modules** section.

If you want a global module to be accessible by external dependencies, you must explicitly make it available. The following options are available to make the services in a global module available externally.

- Add **services="import"** to the module in your **jboss-deployment-structure.xml**
- Add **services="true"** to the global module definition.

```
/subsystem=ee:write-attribute(name=global-modules,value=[{name=module1,services=true}]
```

Or, when adding multiple modules:

```
/subsystem=ee:write-attribute(name=global-modules,value=[{name=module1,services=true},
{name=module2,services=false}]
```

To add a new module to an existing list:

```
/subsystem=ee:list-add(name=global-modules,value={name=module1,services=true})
```

- When defining the global module using the management console, make sure that the value of the *Services* property is **On**.

6.6. CONFIGURE SUBDEPLOYMENT ISOLATION

Each subdeployment in an Enterprise Archive (EAR) is a dynamic module with its own class loader. Subdeployments always have an implicit dependency on the parent module, which gives them access to classes in **EAR/lib**. By default, a subdeployment can access the resources of other subdeployments within that EAR.

If you do not want a subdeployment to be allowed to access classes belonging to other subdeployments, then strict subdeployment isolation can be enabled in JBoss EAP. This setting will affect all deployments.

Enable Subdeployment Module Isolation for All Deployments

Subdeployment isolation can be enabled or disabled using the management console or the management CLI from the **ee** subsystem. By default, subdeployment isolation is set to false, which allows the subdeployments to access resources of other subdeployments within an EAR deployment.

Use the following management CLI to enable EAR subdeployment isolation.

```
/subsystem=ee:write-attribute(name=ear-subdeployments-isolated,value=true)
```

Subdeployments in an EAR will no longer be able to access resources from other subdeployments.

6.7. DEFINE AN EXTERNAL JBOSS EAP MODULE DIRECTORY

The default directory for JBoss EAP modules is **EAP_HOME/modules**. You can specify a different directory for JBoss EAP modules using the **JBOSS_MODULEPATH** variable. Follow the below steps to set this variable in the JBoss EAP startup configuration file.



NOTE

You can also set **JBOSS_MODULEPATH** as an environment variable instead of setting this in the JBoss EAP startup configuration files.

1. Edit the startup configuration file.
 - When running as a standalone server, edit the **EAP_HOME/bin/standalone.conf** file, or **standalone.conf.bat** for Windows Server.
 - When running in a managed domain, edit the **EAP_HOME/bin/domain.conf** file, or **domain.conf.bat** for Windows Server.
2. Set the **JBOSS_MODULEPATH** variable, for example:

```
JBOSS_MODULEPATH="/path/to/modules/directory/"
```

To specify a list of directories use a colon (:) to delimit the list of directories.



NOTE

For Windows Server, use the following syntax to set the **JBOSS_MODULEPATH** variable:

```
set "JBOSS_MODULEPATH /path/to/modules/directory/"
```

To specify a list of directories use a semicolon (;) to delimit the list of directories.

6.8. DYNAMIC MODULE NAMING CONVENTIONS

JBoss EAP loads all deployments as modules, which are named according to the following conventions.

- Deployments of WAR and JAR files are named using the following format:

```
deployment.DEPLOYMENT_NAME
```

For example, **inventory.war** and **store.jar** will have the module names of **deployment.inventory.war** and **deployment.store.jar** respectively.

- Subdeployments within an Enterprise Archive (EAR) are named using the following format:

```
deployment.EAR_NAME.SUBDEPLOYMENT_NAME
```

For example, the subdeployment of **reports.war** within the enterprise archive **accounts.ear** will have the module name of **deployment.accounts.ear.reports.war**.

CHAPTER 7. DEPLOYING APPLICATIONS

JBoss EAP features a range of application deployment and configuration options to cater to both administrators and developers. For administrators, the [management console](#) and the [management CLI](#) offer ideal graphical and command-line interfaces to manage application deployment in a production environment. For developers, the range of application deployment testing options include a configurable file system [deployment scanner](#), the [HTTP API](#), an IDE such as Red Hat CodeReady Studio, and [Maven](#).

When deploying applications, you may want to enable validation for deployment descriptors by setting the **org.jboss.metadata.parser.validate** system property to **true**. This can be done one of the following ways:

- While starting the server.

```
$ EAP_HOME/bin/standalone.sh -Dorg.jboss.metadata.parser.validate=true
```

- By adding it to the server configuration with the following management CLI command.

```
/system-property=org.jboss.metadata.parser.validate:add(value=true)
```

7.1. DEPLOYING APPLICATIONS USING THE MANAGEMENT CLI

Deploying applications using the management CLI gives you the benefit of a single command-line interface with the ability to create and run deployment scripts. You can use this scripting ability to configure specific application deployment and management scenarios. You can manage the deployments for a single server when running as a standalone server, or an entire network of servers when running in a managed domain.

7.1.1. Deploy an Application to a Standalone Server Using the Management CLI

Deploy an Application

From the management CLI, use the **deployment deploy-file** command and specify the path to the application deployment.

```
deployment deploy-file /path/to/test-application.war
```

A successful deployment does not produce any output to the management CLI, but the server log displays deployment messages.

```
WFLYSRV0027: Starting deployment of "test-application.war" (runtime-name: "test-application.war")
WFLYUT0021: Registered web context: /test-application
WFLYSRV0010: Deployed "test-application.war" (runtime-name : "test-application.war")
```

Similarly, you can use the following **deployment** commands:

- Use the **deployment deploy-cli-archive** to deploy the content from a **.cli** archive file. A CLI deployment archive is a **JAR** file with the **.cli** extension. It contains application archives that should be deployed and the CLI script files, **deploy.scr** and **undeploy.scr**, containing commands and operations. One script file, **deploy.scr**, contains the commands and operations that deploy the application archives and set up the environment; the other script file, **undeploy.scr**, contains the commands to undeploy the application archives and clean up the environment.

- Use the **deployment deploy-url** to deploy the content referenced by a URL.

You can also enable the [disabled](#) applications using the **deployment enable** command.

Undeploy an Application

From the management CLI, use the **deployment undeploy** command and specify the deployment name. This will delete the deployment content from the repository. See [Disable an Application](#) for keeping the deployment content when undeploying.

```
deployment undeploy test-application.war
```

A successful undeployment does not produce any output to the management CLI, but the server log displays undeployment messages.

```
WFLYUT0022: Unregistered web context: /test-application
WFLYSRV0028: Stopped deployment test-application.war (runtime-name: test-application.war) in
62ms
WFLYSRV0009: Undeployed "test-application.war" (runtime-name: "test-application.war")
```

Similarly, you can use the **deployment undeploy-cli-archive** to undeploy the content from a **.cli** archive file. You can also undeploy all deployments using a wildcard (*).

```
deployment undeploy *
```

Disable an Application

Disable a deployed application without removing the deployment content from the repository.

```
deployment disable test-application.war
```

You can use the **deployment disable-all** command to disable all the deployments.

```
deployment disable-all
```

Enable an Application

Enable a disabled deployed application.

```
deployment enable test-application.war
```

You can use the **deployment enable-all** command to enable all the deployments.

```
deployment enable-all
```

List Deployments

From the management CLI, use the **deployment info** command to list deployment information.

```
deployment info
```

The output will show details about each deployment, such as the runtime name, status, and whether it is enabled.

```
NAME          RUNTIME-NAME  PERSISTENT  ENABLED  STATUS
helloworld.war helloworld.war true      true    OK
test-application.war test-application.war true      true    OK
```

To display deployment information by name:

```
deployment info helloworld.war
```

You can also list all the deployments using the **deployment list** command.

7.1.2. Deploy an Application in a Managed Domain Using the Management CLI

Deploy an Application

From the management CLI, use the **deployment deploy-file** command and specify the path to the application deployment. You must also specify the server groups to which the application should be deployed.

- To deploy the application to all server groups.

```
deployment deploy-file /path/to/test-application.war --all-server-groups
```

- To deploy the application to specific server groups.

```
deployment deploy-file /path/to/test-application.war --server-groups=main-server-group,other-server-group
```

A successful deployment does not produce any output to the management CLI, but the server log displays deployment messages for each affected server.

```
[Server:server-one] WFLYSRV0027: Starting deployment of "test-application.war" (runtime-name:
"test-application.war")
[Server:server-one] WFLYUT0021: Registered web context: /test-application
[Server:server-one] WFLYSRV0010: Deployed "test-application.war" (runtime-name : "test-
application.war")
```

Similarly, you can use the following **deployment** commands:

- Use the **deployment deploy-cli-archive** command to deploy the content from a **.cli** archive file. A CLI deployment archive is a **JAR** file with the **.cli** extension. It contains application archives that should be deployed and the CLI script files, **deploy.scr** and **undeploy.scr**, containing commands and operations. One script file, **deploy.scr**, contains the commands and operations that deploy the application archives and set up the environment; the other script file, **undeploy.scr**, contains the commands to undeploy the application archives and clean up the environment.
- Use the **deployment deploy-url** command to deploy the content referenced by a URL.

You can also enable the [disabled](#) applications using the **deployment enable** command.

Undeploy an Application

From the management CLI, use the **deployment undeploy** command and specify the deployment name. You must also specify the server groups from which the application should be undeployed. See [Disable an Application](#) for undeploying from specific server groups.

Undeploy the application from all server groups with that deployment.

```
deployment undeploy test-application.war --all-relevant-server-groups
```

A successful undeployment does not produce any output to the management CLI, but the server log displays undeployment messages for each affected server.

```
[Server:server-one] WFLYUT0022: Unregistered web context: /test-application
[Server:server-one] WFLYSRV0028: Stopped deployment test-application.war (runtime-name: test-application.war) in 74ms
[Server:server-one] WFLYSRV0009: Undeployed "test-application.war" (runtime-name: "test-application.war")
```

Similarly, you can use the **deployment undeploy-cli-archive** command to undeploy the content from a **.cli** archive file. You can also undeploy all deployments using a wildcard (*****).

```
deployment undeploy * --all-relevant-server-groups
```

Disable an Application

Disable a deployed application from specific server groups and retain its content in the repository for other server groups with that deployment.

```
deployment disable test-application.war --server-groups=other-server-group
```

You can use the **deployment disable-all** command to disable all the deployments.

```
deployment disable-all --server-groups=other-server-group
```

Enable an Application

Enable a disabled deployed application.

```
deployment enable test-application.war
```

You can use the **deployment enable-all** command to enable all the deployments.

```
deployment enable-all --server-groups=other-server-group
```

List Deployments

From the management CLI, use the **deployment info** command to list deployment information. You can list deployment information by deployment name or by server group.

To display deployment information by name:

```
deployment info helloworld.war
```

The output will list the deployment and its state in each server group.

```
NAME          RUNTIME-NAME
helloworld.war helloworld.war

SERVER-GROUP  STATE
main-server-group enabled
other-server-group added
```

To display deployment information by server group:

```
deployment info --server-group=other-server-group
```


The output will list the deployments and their state for the specified server group.

NAME	RUNTIME-NAME	STATE
helloworld.war	helloworld.war	added
test-application.war	test-application.war	enabled

You can also list all deployments in the domain using the **deployment list** command.

7.2. DEPLOYING APPLICATIONS USING THE MANAGEMENT CONSOLE

Deploying applications using the management console gives you the benefit of a graphical interface that is easy to use. You can see at a glance which applications are deployed to your server or server groups, and you can enable, disable or remove applications from the content repository as required.

7.2.1. Deploy an Application to a Standalone Server Using the Management Console

Deployments can be viewed and managed from the **Deployments** tab of the JBoss EAP management console.

Deploy an Application

Click the Add (+) button. You can choose to deploy an application by *uploading a deployment*, *adding an unmanaged deployment*, or *creating an empty deployment*. Deployments are enabled by default.

- Upload a deployment
Upload an application that will be copied to the server's content repository and managed by JBoss EAP.
- Adding an unmanaged deployment
Specify the location of a deployment. This deployment will not be copied to the server's content repository and will not be managed by JBoss EAP.
- Creating an empty deployment
Create an empty, exploded deployment. You can add files to the deployment after it has been created.

Undeploy an Application

Select the deployment and choose the **Undeploy** option. JBoss EAP removes the deployment from the content repository.

Disable an Application

Select the deployment and choose the **Disable** option to disable the application. This undeploys the deployment, but does not remove it from the content repository.

Replace an Application

Select the deployment and choose the **Replace** option. Select the new version of the deployment, which must have the same name as the original, and click **Finish**. This undeploys and removes the original version of the deployment, and then deploys the new version.

7.2.2. Deploy an Application in a Managed Domain Using the Management Console

From the **Deployments** tab of the JBoss EAP management console, deployments can be viewed and managed by:

- **Content Repository**
All managed and unmanaged deployments are listed in the **Content Repository** section. Deployments can be added and deployed to server groups here.
- **Server Groups**
Deployments that have been deployed to one or more server groups are listed in the **Server Groups** section. Deployments can be enabled and added directly to a server group here.

Add an Application

1. From **Content Repository**, click the **Add** button.
2. Choose to add an application by *uploading a deployment* or *adding an unmanaged deployment*.
3. Follow the prompts to deploy the application.
Note that a deployment must be deployed to a server group before it can be enabled.

Deploy an Application to a Server Group

1. From **Content Repository**, select a deployment and click the **Deploy** button.
2. Select one or more server groups to which this deployment should be deployed.
3. Optionally, select the option to enable the deployment on the selected server groups.

Undeploy an Application from a Server Group

1. From **Server Groups**, select the appropriate server group.
2. Select the desired deployment and click the **Undeploy** button.

Deployments can also be undeployed from multiple server groups at once by selecting the **Undeploy** button for the deployment in **Content Repository**.

Remove an Application

1. If the deployment is still deployed to any server groups, be sure to undeploy the deployment.
2. From **Content Repository**, select the deployment and click the **Remove** button.

This removes the deployment from the content repository.

Disable an Application

1. From **Server Groups**, select the appropriate server group.
2. Select the desired deployment and click the **Disable** button.

This undeploys the deployment, but does not remove it from the content repository.

Replace an Application

1. From **Content Repository**, select the deployment and click the **Replace** button.
2. Select the new version of the deployment, which must have the same name as the original, and click **Replace**.

This undeploys and removes the original version of the deployment, and then deploys the new version.

7.3. DEPLOYING APPLICATIONS USING THE DEPLOYMENT SCANNER

The deployment scanner monitors the deployment directory for applications to deploy. By default, the deployment scanner scans the **`EAP_HOME/standalone/deployments/`** directory every five seconds for changes. Marker files are used to indicate the status of a deployment and to trigger actions against deployments, such as undeploying or redeploying.

While it is recommended to use the management console or management CLI for application deployment in a production environment, deploying using the deployment scanner is provided for the convenience of developers. This allows users to build and test applications in a manner suited for rapid development cycles. Additionally, the deployment scanner should not be used in conjunction with other deployment methods.

The deployment scanner is only available when running JBoss EAP as a standalone server.

7.3.1. Deploy an Application to a Standalone Server Using the Deployment Scanner

The deployment scanner can be configured to allow or disallow automatic deployment of XML, zipped, and exploded content. If automatic deployment is disabled, you must manually create marker files to trigger deployment actions. For more information about the available marker file types and their purposes, see the [Deployment Scanner Marker Files](#) section.

By default, automatic deployment for XML and zipped content is enabled. For details on configuring automatic deployment for each content type, see [Configure the Deployment Scanner](#).



WARNING

Deploying using the deployment scanner is provided for the convenience of developers and is not recommended for use in a production environment. It should also not be used in conjunction with other deployment methods.

Deploy an Application

Copy the content to the deployment folder.

```
$ cp /path/to/test-application.war EAP_HOME/standalone/deployments/
```

If auto-deployment is enabled, this file will be picked up automatically, deployed, and a **.deployed** marker file will be created. If auto-deployment is not enabled, then you will need to manually add a **.dodeploy** marker file to trigger deployment.

```
$ touch EAP_HOME/standalone/deployments/test-application.war.dodeploy
```

Undeploy an Application

Trigger an undeployment by removing the **.deployed** marker file.

```
$ rm EAP_HOME/standalone/deployments/test-application.war.deployed
```

If auto-deployment is enabled, you can also remove the **test-application.war** file, which will trigger the undeployment. Note that this does not apply for exploded deployments.

Redeploy an Application

Create a **.dodeploy** marker file to initiate redeployment.

```
$ touch EAP_HOME/standalone/deployments/test-application.war.dodeploy
```

7.3.2. Configure the Deployment Scanner

The deployment scanner can be configured using the management console or the management CLI. You can configure the deployment scanner's behavior, such as the scan interval, deployment folder location, and autodeployment of certain application file types. You can also disable the deployment scanner entirely.

For details on all available deployment scanner attributes, see the [Deployment Scanner Attributes](#) section.

Use the below management CLI commands to configure the default deployment scanner.

Disable the Deployment Scanner

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-enabled,value=false)
```

This disables the **default** deployment scanner.

Change the Scan Interval

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-interval,value=10000)
```

This updates the scan interval time from **5000** milliseconds (five seconds) to **10000** milliseconds (ten seconds).

Change the Deployment Folder

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=path,value=/path/to/deployments)
```

This changes the location of the deployment folder from the default location of ***EAP_HOME/standalone/deployments*** to ***/path/to/deployments***.

The **path** value will be treated as an absolute path unless the **relative-to** attribute is specified, in which case it will be relative to that path.

Enable the Automatic Deployment of Exploded Content

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-exploded,value=true)
```

This enables the automatic deployment of exploded content, which is disabled by default.

Disable the Automatic Deployment of Zipped Content

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-zipped,value=false)
```

This disables the automatic deployment of zipped content, which is enabled by default.

Disable the Automatic Deployment of XML Content

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-xml,value=false)
```

This disables the automatic deployment of XML content, which is enabled by default.

7.3.3. Define a Custom Deployment Scanner

A new deployment scanner can be added using the management CLI or by navigating to the **Deployment Scanners** subsystem from the **Configuration** tab in the management console. This will define a new directory to scan for deployments. The default deployment scanner monitors **EAP_HOME/standalone/deployments**. See [Configure the Deployment Scanner](#) for details on configuring an existing deployment scanner.

The following management CLI command adds a new deployment scanner that will check **EAP_HOME/standalone/new_deployment_dir** every five seconds for deployments.

```
/subsystem=deployment-scanner/scanner=new-scanner:add(path=new_deployment_dir,relative-to=jboss.server.base.dir,scan-interval=5000)
```



NOTE

The specified directory must already exist or this command will fail with an error.

A new deployment scanner has been defined and the specified directory will be monitored for deployments.

7.4. DEPLOYING APPLICATIONS USING MAVEN

Deploying applications using Apache Maven allows you to easily incorporate deployment to JBoss EAP into your existing development workflow.

You can use Maven to deploy applications to JBoss EAP using the [WildFly Maven Plugin](#), which provides simple operations to deploy and undeploy applications to the application server.

7.4.1. Deploy an Application to a Standalone Server Using Maven

The following instructions show how to deploy and undeploy the JBoss EAP **helloworld** quickstart to a standalone server using Maven.

See [Using the Quickstart Examples](#) in the JBoss EAP *Getting Started Guide* for more information on the JBoss EAP quickstarts.

Deploy an Application

Initialize the WildFly Maven Plugin in your Maven **pom.xml** file. This should already be configured in the JBoss EAP quickstart **pom.xml** files.

```
<plugin>
  <groupId>org.wildfly.plugins</groupId>
  <artifactId>wildfly-maven-plugin</artifactId>
  <version>${version.wildfly.maven.plugin}</version>
</plugin>
```

From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn clean install wildfly:deploy
```

After issuing the Maven command to deploy, the terminal window shows the following output indicating a successful deployment.

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 2.981 s  
[INFO] Finished at: 2015-12-23T15:06:13-05:00  
[INFO] Final Memory: 21M/231M  
[INFO] -----
```

The deployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYSRV0027: Starting deployment of "helloworld.war" (runtime-name: "helloworld.war")  
WFLYUT0021: Registered web context: /helloworld  
WFLYSRV0010: Deployed "helloworld.war" (runtime-name : "helloworld.war")
```

Undeploy an Application

From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn wildfly:undeploy
```

After issuing the Maven command to undeploy, the terminal window shows the following output indicating a successful undeployment.

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 1.237 s  
[INFO] Finished at: 2015-12-23T15:09:10-05:00  
[INFO] Final Memory: 10M/183M  
[INFO] -----
```

The undeployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYUT0022: Unregistered web context: /helloworld  
WFLYSRV0028: Stopped deployment helloworld.war (runtime-name: helloworld.war) in 27ms  
WFLYSRV0009: Undeployed "helloworld.war" (runtime-name: "helloworld.war")
```

7.4.2. Deploy an Application in a Managed Domain Using Maven

The following instructions show how to deploy and undeploy the JBoss EAP **helloworld** quickstart in a managed domain using Maven.

See [Using the Quickstart Examples](#) in the JBoss EAP *Getting Started Guide* for more information on the JBoss EAP quickstarts.

Deploy an Application

When deploying an application in a managed domain, you must specify the server groups to which the application should be deployed. This is configured in the Maven **pom.xml** file.

The following configuration in the **pom.xml** initializes the WildFly Maven Plugin and specifies **main-server-group** as the server group to which the application should be deployed.

```
<plugin>
  <groupId>org.wildfly.plugins</groupId>
  <artifactId>wildfly-maven-plugin</artifactId>
  <version>${version.wildfly.maven.plugin}</version>
  <configuration>
    <domain>
      <server-groups>
        <server-group>main-server-group</server-group>
      </server-groups>
    </domain>
  </configuration>
</plugin>
```

From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn clean install wildfly:deploy
```

After issuing the Maven command to deploy, the terminal window shows the following output indicating a successful deployment.

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.005 s
[INFO] Finished at: 2016-09-02T14:36:17-04:00
[INFO] Final Memory: 21M/226M
[INFO] -----
```

The deployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYSRV0027: Starting deployment of "helloworld.war" (runtime-name: "helloworld.war")
WFLYUT0021: Registered web context: /helloworld
WFLYSRV0010: Deployed "helloworld.war" (runtime-name : "helloworld.war")
```

Undeploy an Application

From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn wildfly:undeploy
```

After issuing the Maven command to undeploy, the terminal window shows the following output indicating a successful undeployment.

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.750 s
```



```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation": "add", "address": {"server-group":
"main-server-group", "deployment": "test-application.war"}, "json.pretty": 1}'
```

3. Deploy the application to the server group.

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation": "deploy", "address": {"server-
group": "main-server-group", "deployment": "test-application.war"}, "json.pretty": 1}'
```

Undeploy an Application

1. Remove the deployment from all server groups to which it is assigned.

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation": "remove", "address": {"server-
group": "main-server-group", "deployment": "test-application.war"}, "json.pretty": 1}'
```

2. Remove the deployment from the content repository.

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation": "remove", "address": {"deployment":
"test-application.war"}, "json.pretty": 1}'
```

7.6. CUSTOMIZING DEPLOYMENT BEHAVIOR

7.6.1. Define a Custom Directory for Deployment Content

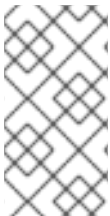
You can define a custom location for JBoss EAP to store deployed content.

Define a Custom Directory for a Standalone Server

By default, deployed content for a standalone server is stored in the ***EAP_HOME/standalone/data/content*** directory. This location can be changed by passing in the ***-Djboss.server.deploy.dir*** argument when starting the server.

```
$ EAP_HOME/bin/standalone.sh -Djboss.server.deploy.dir=/path/to/new_deployed_content
```

The chosen location should be unique among JBoss EAP instances.



NOTE

The ***jboss.server.deploy.dir*** property specifies the directory to be used for storing content that has been deployed using the management console or management CLI. To define a custom deployment directory to be monitored by the deployment scanner, see [Configure the Deployment Scanner](#).

Define a Custom Directory for a Managed Domain

By default, deployed content for a managed domain is stored in the ***EAP_HOME/domain/data/content*** directory. This location can be changed by passing in the ***-Djboss.domain.deployment.dir*** argument when starting the domain.

```
$ EAP_HOME/bin/domain.sh -Djboss.domain.deployment.dir=/path/to/new_deployed_content
```

The chosen location should be unique among JBoss EAP instances.

7.6.2. Control the Order of Deployments

JBoss EAP offers fine-grained control over the order of deployments when the server is started. Strict order of the deployment of applications present in multiple EAR files can be specified along with persistence of the order after a restart.

You can use the **jboss-all.xml** deployment descriptor to declare dependencies between top-level deployments.

For example, if you have an **app.ear** that depends on **framework.ear** being deployed first, then you can create an **app.ear/META-INF/jboss-all.xml** file as shown below.

```
<jboss xmlns="urn:jboss:1.0">
  <jboss-deployment-dependencies xmlns="urn:jboss:deployment-dependencies:1.0">
    <dependency name="framework.ear" />
  </jboss-deployment-dependencies>
</jboss>
```



NOTE

You can use the deployment's runtime name as the dependency name in the **jboss-all.xml** file.

This ensures that **framework.ear** is deployed before **app.ear**.



IMPORTANT

If you create a **jboss-all.xml** file in **app.ear** and you do not deploy **framework.ear**, the server attempts to deploy **app.ear** and fails.

7.6.3. Override Deployment Content

A *deployment overlay* can be used to overlay content into an existing deployment without physically modifying the contents of the deployment archive. It allows you to override deployment descriptors, library JAR files, classes, JSP pages, and other files at runtime without rebuilding the archive.

This can be useful if you need to adapt a deployment for different environments that need different configurations or settings. For example, when moving a deployment through the application lifecycle from development, to testing, to stage, and finally into production, you might want to swap deployment descriptors, modify static web resources to change the branding of the application, or even replace JAR libraries with different versions depending on the target environment. It is also a useful feature for installations that need to change a configuration but can not modify or crack an archive due to policy or security restrictions.

When defining a deployment overlay, you specify the file on a file system that will replace the file in the deployment archive. You must also specify which deployments should be affected by the deployment overlay. Any affected deployments must be redeployed in order for the changes to take effect.

Parameters

You can use any of the following parameters to configure your deployment overlay:

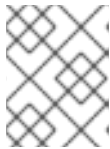
- **name**:: The name of the deployment overlay.
- **content**:: A comma-separated list that maps the file on the file system to the file in the archive that it replaces. The format for each entry is **ARCHIVE_PATH=FILESYSYEM_PATH**.
- **deployments**:: Comma-separated list of deployments to which this overlay is linked.
- **redploy-affected**:: Redeploys all affected deployments.

For full usage details, execute **deployment-overlay --help**.

Procedure

1. Use the **deployment-overlay add** management CLI command to add a deployment overlay:

```
deployment-overlay add --name=new-deployment-overlay --content=WEB-INF/web.xml=/path/to/other/web.xml --deployments=test-application.war --redploy-affected
```



NOTE

In a managed domain, specify the applicable server groups by using **--server-groups** or specify all server groups with **--all-server-groups**.

2. After you created a deployment overlay, you can add content to an existing overlay, link the overlay to a deployment, or remove the overlay.
3. Optional: You can specify an overlay configuration to link to an external directory that contains static web resources, such as HTML, images, or videos, in the **<overlay>** element. The **<overlay>** element is located in the application **jboss-web.xml** file. With this configuration, you do not need to repackage the application.

The following example shows system property substitution in the **<overlay>** element, where `${example.path.to.overlay}` defines the `/PATH/TO/STATIC/WEB/CONTENT` location.

Example: **<overlay>** element in a **jboss-web.xml** file

```
<jboss-web>
  <overlay>${example.path.to.overlay}</overlay>
</jboss-web>
```

You can specify a system property in the **<overlay>** element if **jboss-descriptor-property-replacement** is set to **true**, which is the default value for the descriptor property.

To configure **jboss-descriptor-property-replacement**, use the following management CLI command:

```
/subsystem=ee:write-attribute(name=jboss-descriptor-property-replacement,value=true)
```

This command adds the following XML content to the **ee** subsystem in the JBoss EAP configuration:

```
<subsystem xmlns="urn:jboss:domain:ee:4.0">
  <jboss-descriptor-property-replacement>true</jboss-descriptor-property-replacement>
</subsystem>
```

7.6.4. Using Rollout Plans

About Rollout Plans

In a managed domain, operations targeted at domain or host level resources can potentially impact multiple servers. Such operations can include a roll out plan detailing the sequence in which the operation would be applied to the servers, as well as the policies for detailing whether the operation could be reverted if it fails to execute successfully on some servers. If no rollout plan is specified, the [default rollout plan](#) is used.

Below is an example rollout plan involving five server groups. Operations can be applied to server groups serially, **in-series**, or concurrently, **concurrent-groups**. The syntax is described in more detail in [Rollout Plan Syntax](#).

```
{ "my-rollout-plan" => { "rollout-plan" => {
  "in-series" => [
    { "concurrent-groups" => {
      "group-A" => {
        "max-failure-percentage" => "20",
        "rolling-to-servers" => "true"
      },
      "group-B" => undefined
    } },
    { "server-group" => { "group-C" => {
      "rolling-to-servers" => "false",
      "max-failed-servers" => "1"
    } } },
    { "concurrent-groups" => {
      "group-D" => {
        "max-failure-percentage" => "20",
        "rolling-to-servers" => "true"
      },
      "group-E" => undefined
    } }
  ],
  "rollback-across-groups" => "true"
} }
```

Looking at the example above, applying the operation to the servers in the domain is done in three phases. If the policy for any server group triggers a rollback of the operation across the server group, all other server groups will be rolled back as well.

1. Server groups *group-A* and *group-B* will have the operation applied concurrently. The operation will be applied to the servers in *group-A* in series, while all servers in *group-B* will handle the operation concurrently. If more than 20% of the servers in *group-A* fail to apply the operation, it will be rolled back across that group. If any servers in *group-B* fail to apply the operation it will be rolled back across that group.
2. Once all servers in *group-A* and *group-B* are complete, the operation will be applied to the servers in *group-C*. Those servers will handle the operation concurrently. If more than one server in *group-C* fails to apply the operation it will be rolled back across that group.
3. Once all servers in *group-C* are complete, server groups *group-D* and *group-E* will have the operation applied concurrently. The operation will be applied to the servers in *group-D* in series, while all servers in *group-E* will handle the operation concurrently. If more than 20% of the servers in *group-D* fail to apply the operation, it will be rolled back across that group. If any servers in *group-E* fail to apply the operation it will be rolled back across that group.

Rollout Plan Syntax

You can specify a rollout plan in either of the following ways.

- By defining the rollout plan in the **deploy** command operation headers. See [Deploy Using a Rollout Plan](#) for details.
- By storing the rollout plan using the **rollout-plan** command and then referencing the plan name in the **deploy** command operation headers. See [Deploy Using a Stored Rollout Plan](#) for details.

Although each method has a different initial command, both methods use the **rollout** operation header to define the rollout plan. This uses the following syntax.

```
rollout (id=PLAN_NAME | SERVER_GROUP_LIST) [rollback-across-groups]
```

- **PLAN_NAME** is the name for the rollout plan that was stored using the **rollout-plan** command.
- **SERVER_GROUP_LIST** is the list of server groups. Use a comma (,) to separate multiple server groups to indicate that operations should be performed on each server group sequentially. Use a caret (^) separator to indicate that operations should be performed on each server group concurrently.
 - For each server group, set any of the following policies in parentheses. Use a comma to separate multiple policies.
 - **rolling-to-servers**: A boolean that, if set to **true**, applies the operation to each server in the group in series. If the value is **false** or not specified, the operation will be applied to the servers in the group concurrently.
 - **max-failed-servers**: An integer which takes the maximum number of servers in the group that can fail to apply the operation before it should be reverted on all servers in the group. The default value if not specified is **0**, meaning that a failure on any server will trigger rollback across the group.
 - **max-failure-percentage**: An integer between **0** and **100** that represents the maximum percentage of the total number of servers in the group that can fail to apply the operation before it should be reverted on all servers in the group. The default value if not specified is **0**, meaning that a failure on any server will trigger rollback across the group.



NOTE

If both **max-failed-servers** and **max-failure-percentage** are set to non-zero values, **max-failure-percentage** takes precedence.

- **rollback-across-groups**: A boolean that indicates whether the need to rollback the operation on all the servers in one server group triggers a rollback across all the server groups. This defaults to **false**.

Deploy Using a Rollout Plan

You can provide the full details of a rollout plan directly to the **deploy** command by passing the **rollout** settings into the **headers** argument. See the [Rollout Plan Syntax](#) for more information on the format.

The following management CLI command deploys an application to the **main-server-group** server group using a deployment plan that specifies **rolling-to-servers=true** for serial deployment.

```
deploy /path/to/test-application.war --server-groups=main-server-group --headers={rollout main-
server-group(rolling-to-servers=true)}
```

Deploy Using a Stored Rollout Plan

Since rollout plans can be complex, you have the option to store the details of a rollout plan. This allows you to reference the rollout plan name when you want to use it instead of requiring the full details of the rollout plan each time.

1. Use the **rollout-plan** management CLI command to store a rollout plan. See the [Rollout Plan Syntax](#) for more information on the format.

```
rollout-plan add --name=my-rollout-plan --content={rollout main-server-group(rolling-to-
servers=false,max-failed-servers=1),other-server-group(rolling-to-servers=true,max-failure-
percentage=20) rollback-across-groups=true}
```

This creates the following deployment plan.

```
"rollout-plan" => {
  "in-series" => [
    {"server-group" => {"main-server-group" => {
      "rolling-to-servers" => false,
      "max-failed-servers" => 1
    }},
    {"server-group" => {"other-server-group" => {
      "rolling-to-servers" => true,
      "max-failure-percentage" => 20
    }}}
  ],
  "rollback-across-groups" => true
}
```

2. Specify the stored rollout plan name when deploying the application.
The following management CLI command deploys an application to all server groups using the **my-rollout-plan** stored rollout plan.

```
deploy /path/to/test-application.war --all-server-groups --headers={rollout id=my-rollout-plan}
```

Remove a Stored Rollout Plan

You can remove a stored rollout plan using the **rollout-plan** management CLI command by specifying the name of the rollout plan to remove.

```
rollout-plan remove --name=my-rollout-plan
```

Default Rollout Plan

All operations that impact multiple servers will be executed with a rollout plan. If no rollout plan is specified in the operation request, a default rollout plan will be generated. The plan will have the following characteristics.

- There will only be a single high-level phase. All server groups affected by the operation will have the operation applied concurrently.
- Within each server group, the operation will be applied to all servers concurrently.
- Failure on any server in a server group will cause rollback across the group.

- Failure of any server group will result in rollback of all other server groups.

7.7. MANAGING EXPLODED DEPLOYMENTS

Prior to JBoss EAP 7.1, you could only manage exploded deployments by manipulating files on the file system. Starting with JBoss EAP 7.1, you can manage exploded deployments using the management interfaces. This allows you to change the contents of an exploded application without deploying a new version of the application.



NOTE

Updates to static files in a deployment, such as JavaScript and CSS files, take effect immediately. Changes to other files, such as Java classes, might require an application redeployment for the changes to take effect.

You can either start with an [empty deployment](#) or [explode an existing archive deployment](#) and then [add or remove content](#).

See [Viewing Deployment Content](#) to browse the files in a deployment or read the contents of the files.

Create an Empty Exploded Deployment

You can create an empty exploded deployment to which you can later add content as necessary. Use the following management CLI command to create an empty exploded deployment.

```
/deployment=DEPLOYMENT_NAME.war:add(content=[{empty=true}])
```

The **empty=true** option is required to confirm that you intended to create an empty deployment.

Explode an Existing Archive Deployment

You can explode an existing archive deployment to be able to update its contents. Note that a [deployment must be disabled](#) before it can be exploded. Use the following management CLI command to explode a deployment.

```
/deployment=ARCHIVE_DEPLOYMENT_NAME.ear:explode
```

You can now [add or remove content](#) from this deployment.



NOTE

You can also explode an existing archive deployment from the management console. From the **Deployments** tab, select the deployment and select the **Explode** drop down option.

Add Content to an Exploded Deployment

To add content to a deployment, use the **add-content** management CLI operation. Provide the path to the location in the deployment where the content should be added, and provide the content to be uploaded. The content to upload can be provided as a local file stream, URL, hash of content that already exists in the JBoss EAP content repository, or a byte array of the content. The following management CLI command uses the **input-stream-index** option to upload the contents of a local file to the deployment.

```
/deployment=DEPLOYMENT_NAME.war:add-content(content=[{target-path=/path/to/FILE_IN_DEPLOYMENT, input-stream-index=/path/to/LOCAL_FILE_TO_UPLOAD}])
```



NOTE

When adding content to a deployment using the **add-content** operation, content in the deployment is overwritten by default. You can change this behavior by setting the **overwrite** option to **false**.

Remove Content from an Exploded Deployment

To remove content from a deployment, use the **remove-content** management CLI operation and provide the path of the content in the deployment to remove.

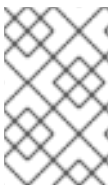
```
/deployment=DEPLOYMENT_NAME.war:remove-content(paths=[/path/to/FILE_1, /path/to/FILE_2])
```

7.8. VIEWING DEPLOYMENT CONTENT

You can [browse information](#) about files in a managed deployment and [read the contents](#) of the files using the JBoss EAP management interfaces.

7.8.1. Browse Files in a Deployment

Use the **browse-content** operation to view the files and directories in a managed deployment. Provide no arguments to return the entire deployment structure or use the **path** argument to provide the path to a specific directory.



NOTE

You can also browse contents of a deployment from the management console by navigating to the **Deployments** tab, selecting the deployment, and selecting **View** from the drop down.

```
/deployment=helloworld.war:browse-content(path=META-INF/)
```

This displays the files and directories in the **META-INF/** directory of the **helloworld.war** deployment.

```
{
  "outcome" => "success",
  "result" => [
    {
      "path" => "MANIFEST.MF",
      "directory" => false,
      "file-size" => 827L
    },
    {
      "path" => "maven/org.jboss.eap.quickstarts/helloworld/pom.properties",
      "directory" => false,
      "file-size" => 106L
    },
    {
      "path" => "maven/org.jboss.eap.quickstarts/helloworld/pom.xml",
      "directory" => false,
      "file-size" => 2713L
    },
    {
      "path" => "maven/org.jboss.eap.quickstarts/helloworld/",
```



```

        "directory" => true
      },
      {
        "path" => "maven/org.jboss.eap.quickstarts/",
        "directory" => true
      },
      {
        "path" => "maven/",
        "directory" => true
      },
      {
        "path" => "INDEX.LIST",
        "directory" => false,
        "file-size" => 251L
      }
    ]
  }
}

```

You can also specify the following arguments to the **browse-content** operation.

archive

Whether to only return archive files.

depth

Specify the depth of files to return.

7.8.2. Read Deployment Content

You can read the contents of a file in a managed deployment using the **read-content** operation. Provide no arguments to return the entire deployment or use the **path** argument to provide the path to a specific file. For example:

```
/deployment=helloworld.war:read-content(path=META-INF/MANIFEST.MF)
```

This returns a file stream, which can be [displayed in the management CLI](#) or [saved to the file system](#).

```

{
  "outcome" => "success",
  "result" => {"uuid" => "24ba8e06-21bd-4505-b4d4-bdfb16451b95"},
  "response-headers" => {"attached-streams" => [{
    "uuid" => "24ba8e06-21bd-4505-b4d4-bdfb16451b95",
    "mime-type" => "text/plain"
  }]}
}

```

7.8.2.1. Display the Contents of a File

Use the **attachment display** command to read the contents of the **MANIFEST.MF** file.

```
attachment display --operation=/deployment=helloworld.war:read-content(path=META-INF/MANIFEST.MF)
```

This displays the contents of the **MANIFEST.MF** file from the **helloworld.war** deployment to the management CLI.

```
ATTACHMENT 8af87836-2abd-423a-8e44-e731cc57bd80:
Manifest-Version: 1.0
Implementation-Title: Quickstart: helloworld
Implementation-Version: 7.3.0.GA
Java-Version: 1.8.0_131
Built-By: username
Scm-Connection: scm:git:git@github.com:jboss/jboss-parent-pom.git/quic
kstart-parent/helloworld
Specification-Vendor: JBoss by Red Hat
...
```

7.8.2.2. Save the Contents of a File

Use the **attachment save** command to save the contents of the **MANIFEST.MF** file to the file system.

```
attachment save --operation=/deployment=helloworld.war:read-content(path=META-
INF/MANIFEST.MF) --file=/path/to/MANIFEST.MF
```

This saves the **MANIFEST.MF** file from the **helloworld.war** deployment to the file system at **path/to/MANIFEST.MF**. If you do not specify a file path using the **--file** argument, the file will be named using its unique attachment ID and saved in the working directory of the management CLI, which by default is **EAP_HOME/bin/**.

CHAPTER 8. DOMAIN MANAGEMENT

This section discusses concepts and configuration specific to the managed domain operating mode.

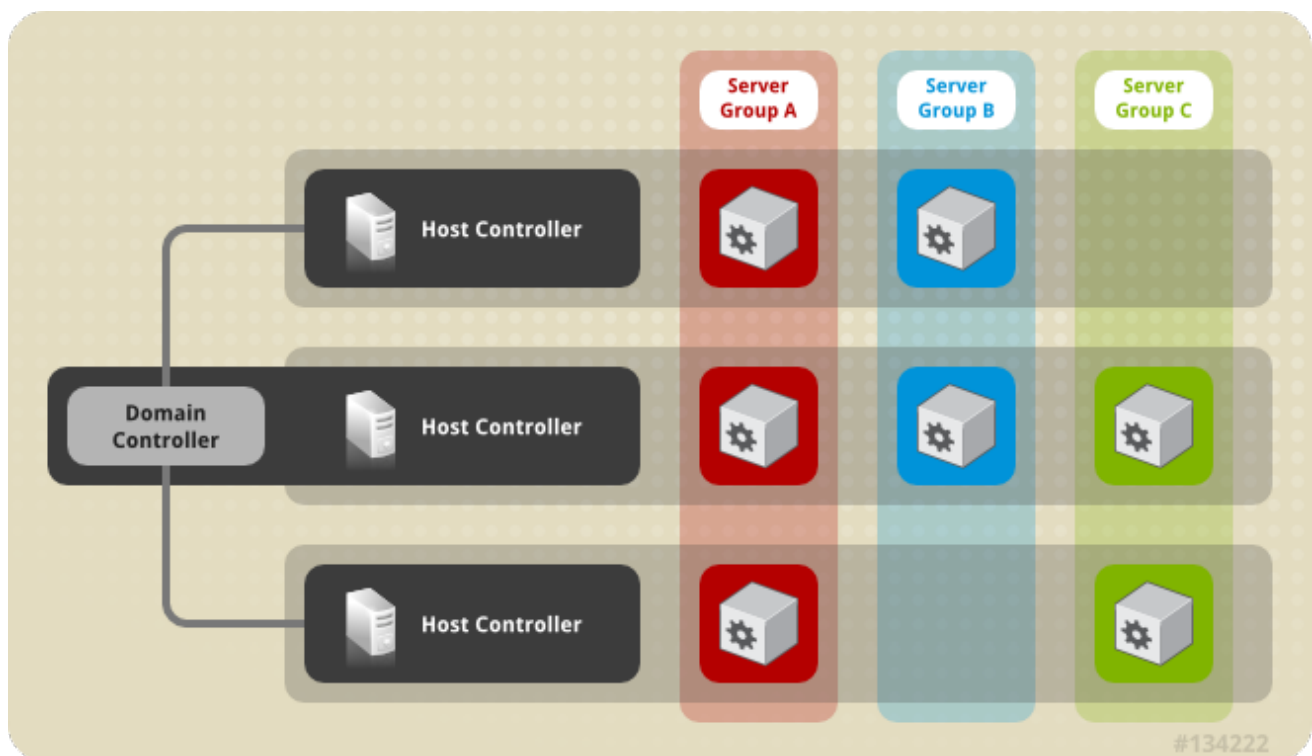
For information on securing a managed domain, see the [Securing a Managed Domain](#) section of JBoss EAP *How to Configure Server Security*.

8.1. ABOUT MANAGED DOMAINS

The managed domain operating mode allows for the management of multiple JBoss EAP instances from a single control point.

Centrally-managed JBoss EAP server collections are known as members of a domain. All JBoss EAP instances in a domain share a common management policy.

A domain consists of one domain controller, one or more host controllers, and zero or more server groups per host.



A [domain controller](#) is the central point from which the domain is controlled. It ensures that each server is configured according to the management policy of the domain. The domain controller is also a host controller.

A [host controller](#) is a physical or virtual host that interacts with the domain controller to control the lifecycle of the application server instances running on its host and to assist the domain controller to manage them. Each host can contain multiple server groups.

A [server group](#) is a set of [server](#) instances which have JBoss EAP installed on them and are managed and configured as one. The domain controller manages the configuration of and applications deployed onto server groups. Consequently, each server in a server group shares the same configuration and deployments.

Host controllers are tied to specific physical, or virtual, hosts. You can run multiple host controllers on the same hardware if you use different configurations, ensuring their ports and other resources do not conflict. It is possible for a domain controller, a single host controller, and multiple servers to run within

the same JBoss EAP instance, on the same physical system.

8.1.1. About the Domain Controller

A domain controller is the JBoss EAP server instance that acts as a central management point for a domain. One host controller instance is configured to act as a domain controller.

The primary responsibilities of the domain controller are:

- Maintain the domain's central management policy.
- Ensure all host controllers are aware of its current contents.
- Assist the host controllers in ensuring that all running JBoss EAP server instances are configured in accordance with this policy.

By default, the central management policy is stored in the **EAP_HOME/domain/configuration/domain.xml** file. This file is required in this directory of the host controller that is set to run as the domain controller.

The **domain.xml** file contains profile configurations available for use by the servers in the domain. A [profile](#) contains the detailed settings of the various subsystems available in that profile. The domain configuration also includes the definition of socket groups and the server group definitions.



NOTE

It is possible for a JBoss EAP 7 domain controller to administer JBoss EAP 6 hosts and servers, as long as the hosts and servers are running JBoss EAP 6.2 or later. For more information, see [Configure a JBoss EAP 7.x Domain Controller to Administer JBoss EAP 6 Instances](#).

For more information, see the [Start a Managed Domain](#) and [Domain Controller Configuration](#) sections.

8.1.2. About Host Controllers

The primary responsibility of a host controller is server management. It delegates domain management tasks and is responsible for starting and stopping the individual application server processes that run on its host.

It interacts with the domain controller to help manage the communication between the servers and the domain controller. Multiple host controllers of a domain can interact with only a single domain controller. Hence, all the host controllers and server instances running on a single domain mode have a single domain controller and must belong to the same domain.

By default, each host controller reads its configuration from the **EAP_HOME/domain/configuration/host.xml** file located in the unzipped JBoss EAP installation file on its host's file system. The **host.xml** file contains the following configuration information that is specific to the particular host:

- The names of the server instances meant to run from this installation.
- Configurations specific to the local physical installation. For example, named interface definitions declared in **domain.xml** can be mapped to an actual machine-specific IP address in **host.xml**. And abstract path names in **domain.xml** can be mapped to actual file system paths in **host.xml**.

- Any of the following configurations:
 - How the host controller contacts the domain controller to register itself and access the domain configuration.
 - How to find and contact a remote domain controller.
 - Whether the host controller is to act as the domain controller

For more information, see the [Start a Managed Domain](#) and [Host Controller Configuration](#) sections.

8.1.3. About Process Controllers

A process controller is a small, lightweight process that is responsible for spawning the host controller process and monitoring its lifecycle. If the host controller crashes, the process controller will restart it. It also starts server processes as directed by the host controller; however, it will not automatically restart server processes that crash.

The process controller logs to the ***EAP_HOME/domain/log/process-controller.log*** file. You can set JVM options for the process controller in the ***EAP_HOME/bin/domain.conf*** file using the **PROCESS_CONTROLLER_JAVA_OPTS** variable.

8.1.4. About Server Groups

A server group is a collection of server instances that are managed and configured as one. In a managed domain, every application server instance belongs to a server group, even if it is the only member. The server instances in a group share the same profile configuration and deployed content.

A domain controller and a host controller enforce the standard configuration on all server instances of every server group in its domain.

A domain can consist of multiple server groups. Different server groups can be configured with different profiles and deployments. For example, a domain can be configured with different server tiers providing different services.

Different server groups can also have the same profile and deployments. This can, for example, allow for rolling application upgrades where the application is upgraded on one server group and then updated on a second server group, avoiding a complete service outage.

For more information, see the [Configuring Server Groups](#) section.

8.1.5. About Servers

A server represents an application server instance. In a managed domain, all server instances are members of a server group. The host controller launches each server instance in its own JVM process.

For more information, see the [Configuring Servers](#) section.

8.2. NAVIGATING DOMAIN CONFIGURATIONS

JBoss EAP provides scalable management interfaces to support both small and large-scale managed domains.

Management Console

The JBoss EAP management console provides a graphical view of your domain and allows you to easily manage hosts, servers, deployments, and profiles for your domain.

Configuration

From the **Configuration** tab, you can configure the subsystems for each profile used in your domain. Different server groups in your domain may use different profiles depending the capabilities needed. Once you select the desired profile, all available subsystems for that profile are listed. For more information on configuring profiles, see [Managing JBoss EAP Profiles](#).

Runtime

From the **Runtime** tab, you can manage servers and server groups as well as host configuration. You can browse the domain by host or by server group.

From **Hosts**, you can configure host properties and JVM settings as well as add and configure servers for that host.

From **Server Groups**, you can add new server groups and configure properties and JVM settings as well as add and configure servers for that server group. You can perform operations such as starting, stopping, suspending, and reloading all servers in the selected server group.

From either **Hosts** or **Server Groups**, you can add new servers and configure server properties and JVM settings. You can perform operations such as starting, stopping, suspending, and reloading the selected server. You can also view runtime information, such as JVM usage, server logs, and subsystem-specific information.

From **Topology**, you can see an overview and view detailed information for the hosts, server groups, and servers in your domain. You can perform operations on each of them, such as reloading or suspending.

Deployments

From the **Deployments** tab, you can add and deploy deployments to server groups. You can view all deployments in the content repository or view deployments deployed to a particular server group.

For more information on deploying applications using the management console, see [Deploy an Application in a Managed Domain](#)

Management CLI

The JBoss EAP management CLI provides a command-line interface to manage hosts, servers, deployments and profiles for your domain.

Subsystem configuration can be accessed once the appropriate profile is selected.

```
/profile=PROFILE_NAME/subsystem=SUBSYSTEM_NAME:read-resource(recursive=true)
```



NOTE

Instructions and examples throughout this guide may contain management CLI commands for subsystem configuration that apply when running as a standalone server, for example:

```
/subsystem=datasources/data-source=ExampleDS:read-resource
```

To adjust these management CLI commands to be run in a managed domain, you must specify the appropriate profile to configure, for example:

```
/profile=default/subsystem=datasources/data-source=ExampleDS:read-resource
```

After specifying the appropriate host, you can configure host settings and perform operations on servers on that host.

```
/host=HOST_NAME/server=SERVER_NAME:read-resource
```

After specifying the appropriate host, you can configure servers for that host.

```
/host=HOST_NAME/server-config=SERVER_NAME:write-attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

After specifying the appropriate server group, you can configure server group settings and perform operations on all servers in the selected server group.

```
/server-group=SERVER_GROUP_NAME:read-resource
```

You can deploy applications in a managed domain by using the **deploy** management CLI command and specifying the appropriate server groups. For instructions, see [Deploy an Application in a Managed Domain](#).

8.3. LAUNCHING A MANAGED DOMAIN

8.3.1. Start a Managed Domain

Domain and host controllers can be started using the **domain.sh** or **domain.bat** script provided with JBoss EAP. For a complete listing of all available startup script arguments and their purposes, use the **--help** argument or see the [Server Runtime Arguments](#) section.

The domain controller must be started before any slave servers in any server groups in the domain. Start the domain controller first, then start any other associated host controllers in the domain.

Start the Domain Controller

Start the domain controller with the **host-master.xml** configuration file, which is preconfigured for a dedicated domain controller.

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml
```

Depending on your domain setup, you will need to make additional configurations to allow host controllers to connect. Also see the following example domain setups:

- [Set Up a Managed Domain on a Single Machine](#)
- [Set Up a Managed Domain on Two Machines](#)

Start a Host Controller

Start the host controller with the **host-slave.xml** configuration file, which is preconfigured for a slave host controller.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

Depending on your domain setup, you will need to make additional configurations connect to, and not conflict with, the domain controller. Also see the following example domain setups:

- [Set Up a Managed Domain on a Single Machine](#)
- [Set Up a Managed Domain on Two Machines](#)

8.3.2. Domain Controller Configuration

You must configure one host in the domain as the domain controller.



IMPORTANT

It is not supported to configure multiple domain or host controllers on the same machine when using the RPM installation method to install JBoss EAP.

Configure a host as the domain controller by adding the **<local/>** element in the **<domain-controller>** declaration. The **<domain-controller>** should include no other content.

```
<domain-controller>  
  <local/>  
</domain-controller>
```

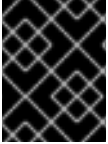
The host acting as the domain controller must expose a management interface that is accessible to other hosts in the domain. The HTTP interface is the standard management interface.

```
<management-interfaces>  
  <http-interface security-realm="ManagementRealm" http-upgrade-enabled="true">  
    <socket interface="management" port="${jboss.management.http.port:9990}"/>  
  </http-interface>  
</management-interfaces>
```

The sample minimal domain controller configuration file, **EAP_HOME/domain/configuration/host-master.xml**, includes these configuration settings.

8.3.3. Host Controller Configuration

A host controller must be configured to connect to the domain controller so that the host controller can register itself with the domain.



IMPORTANT

It is not supported to configure multiple domain or host controllers on the same machine when using the RPM installation method to install JBoss EAP.

Use the **<domain-controller>** element of the configuration to configure a connection to the domain controller.

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9990}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

The sample minimal host controller configuration file, **EAP_HOME/domain/configuration/host-slave.xml**, includes the configuration settings to connect to a domain controller. The configuration assumes that you provide the **jboss.domain.master.address** property when starting the host controller.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Djboss.domain.master.address=IP_ADDRESS
```

For more information on domain controller discovery, see the [Domain Controller Discovery and Failover](#) section.

Depending on your domain setup, you may also need to provide authentication for the host controller to be authenticated by the domain controller. See [Set Up a Managed Domain on Two Machines](#) for details on generating a management user with a secret value and updating the host controller configuration with that value.

8.3.4. Configure the Name of a Host

Every host running in a managed domain must have a unique host name. To ease administration and allow for the use of the same host configuration files on multiple hosts, the server uses the following precedence for determining the host name.

1. If set, the host element name attribute in the **host.xml** configuration file.
2. The value of the **jboss.host.name** system property.
3. The value that follows the final period (.) character in the **jboss.qualified.host.name** system property, or the entire value if there is no final period (.) character.
4. The value that follows the period (.) character in the **HOSTNAME** environment variable for POSIX-based operating systems, the **COMPUTERNAME** environment variable for Microsoft Windows, or the entire value if there is no final period (.) character.

A host controller's name is configured in the **host** element at the top of the relevant **host.xml** configuration file, for example:

```
<host xmlns="urn:jboss:domain:8.0" name="host1">
```

Use the following procedure to update the host name using the management CLI.

1. Start the JBoss EAP host controller.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

2. Launch the management CLI, connecting to the domain controller.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --
controller=DOMAIN_CONTROLLER_IP_ADDRESS
```

3. Use the following command to set a new host name.

```
/host=EXISTING_HOST_NAME:write-attribute(name=name,value=NEW_HOST_NAME)
```

This modifies the host name attribute in the **host-slave.xml** file as follows:

```
<host name="NEW_HOST_NAME" xmlns="urn:jboss:domain:8.0">
```

4. Reload the host controller in order for the changes to take effect.

```
reload --host=EXISTING_HOST_NAME
```

If a host controller does not have a name set in the configuration file, you can also pass in the host name at runtime.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -Djboss.host.name=HOST_NAME
```

8.4. MANAGING SERVERS

8.4.1. Configure Server Groups

The following is an example of a server group definition:

```
<server-group name="main-server-group" profile="full">
  <jvm name="default">
    <heap size="64m" max-size="512m"/>
  </jvm>
  <socket-binding-group ref="full-sockets"/>
  <deployments>
    <deployment name="test-application.war" runtime-name="test-application.war"/>
    <deployment name="helloworld.war" runtime-name="helloworld.war" enabled="false"/>
  </deployments>
</server-group>
```

Server groups can be configured using the management CLI or from the management console **Runtime** tab.

Add a Server Group

The following management CLI command can be used to add a server group.

```
/server-group=SERVER_GROUP_NAME:add(profile=PROFILE_NAME,socket-binding-  
group=SOCKET_BINDING_GROUP_NAME)
```

Update a Server Group

The following management CLI command can be used to update server group attributes.

```
/server-group=SERVER_GROUP_NAME:write-attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

Remove a Server Group

The following management CLI command can be used to remove a server group.

```
/server-group=SERVER_GROUP_NAME:remove
```

Server Group Attributes

A server group requires the following attributes:

- **name**: The server group name.
- **profile**: The server group profile name.
- **socket-binding-group**: The default socket binding group used for servers in the group. This can be overridden on a per-server basis.

A server group includes the following optional attributes:

- **management-subsystem-endpoint**: Set to **true** to have servers belonging to the server group connect back to the host controller using the endpoint from their **remoting** subsystem. The **remoting** subsystem must be present for this to work.
- **socket-binding-default-interface**: The socket binding group default interface for this server.
- **socket-binding-port-offset**: The default offset to be added to the port values given by the socket binding group.
- **deployments**: The deployment content to be deployed on the servers in the group.
- **jvm**: The default JVM settings for all servers in the group. The host controller merges these settings with any other configuration provided in **host.xml** to derive the settings used to launch the server's JVM.
- **deployment-overlays**: Links between a defined deployment overlay and deployments in this server group.
- **system-properties**: The system properties to be set on servers in the group.

8.4.2. Configure Servers

The default **host.xml** configuration file defines three servers:

```
<servers>  
  <server name="server-one" group="main-server-group">  
  </server>  
  <server name="server-two" group="main-server-group" auto-start="true">  
    <socket-bindings port-offset="150"/>  
  </server>
```

```
<server name="server-three" group="other-server-group" auto-start="false">
  <socket-bindings port-offset="250"/>
</server>
</servers>
```

A server instance named **server-one** is associated with **main-server-group** and inherits the subsystem configuration and socket bindings specified by that server group. A server instance named **server-two** is also associated with **main-server-group**, but also defines a socket binding **port-offset** value, so as not to conflict with the port values used by **server-one**. A server instance named **server-three** is associated with **other-server-group** and uses that group's configurations. It also defines a **port-offset** value and sets **auto-start** to **false** so that this server does not start when the host controller starts.

Servers can be configured using the management CLI or from the management console **Runtime** tab.

Add a Server

The following management CLI command can be used to add a server.

```
/host=HOST_NAME/server-config=SERVER_NAME:add(group=SERVER_GROUP_NAME)
```

Update a Server

The following management CLI command can be used to update server attributes.

```
/host=HOST_NAME/server-config=SERVER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

Remove a Server

The following management CLI command can be used to remove a server.

```
/host=HOST_NAME/server-config=SERVER_NAME:remove
```

Server Attributes

A server requires the following attributes:

- **name**: The name of the server.
- **group**: The name of a server group from the domain model.

A server includes the following optional attributes:

- **auto-start**: Whether or not this server should be started when the host controller starts.
- **socket-binding-group**: The socket binding group to which this server belongs.
- **socket-binding-port-offset**: An offset to be added to the port values given by the socket binding group for this server.
- **update-auto-start-with-server-status**: Update the **auto-start** attribute with the status of the server.
- **interface**: A list of fully-specified named network interfaces available for use on the server.
- **jvm**: The JVM settings for this server. If not declared, the settings are inherited from the parent server group or host.
- **path**: A list of named file system paths.

- **system-property:** A list of system properties to set on this server.

8.4.3. Start and Stop Servers

You can perform operations on servers, such as starting, stopping, and reloading, from the management console by navigating to the **Runtime** tab and selecting the appropriate host or server group.

See the below commands for performing these operations using the management CLI.

Start Servers

You can start a single server on a particular host.

```
/host=HOST_NAME/server-config=SERVER_NAME:start
```

You can start all servers in a specified server group.

```
/server-group=SERVER_GROUP_NAME:start-servers
```

Stop Servers

You can stop a single server on a particular host.

```
/host=HOST_NAME/server-config=SERVER_NAME:stop
```

You can stop all servers in a specified server group.

```
/server-group=SERVER_GROUP_NAME:stop-servers
```

Reload Servers

You can reload a single server on a particular host.

```
/host=HOST_NAME/server-config=SERVER_NAME:reload
```

You can reload all servers in a specified server group.

```
/server-group=SERVER_GROUP_NAME:reload-servers
```

Kill Servers

You can kill all server processes in a specified server group.

```
/server-group=SERVER_GROUP_NAME:kill-servers
```

8.5. DOMAIN CONTROLLER DISCOVERY AND FAILOVER

When setting up a managed domain, each host controller must be configured with information needed to contact the domain controller. In JBoss EAP, each host controller can be configured with [multiple options](#) for finding the domain controller. Host controllers iterate through the list of options until one succeeds.

A backup host controller can be [promoted to domain controller](#) if there is a problem with the primary domain controller. This allows host controllers to automatically fail over to the new domain controller once it has been promoted.

8.5.1. Configure Domain Discovery Options

The following is an example of how to configure a host controller with multiple options for finding the domain controller.

Example: A Host Controller with Multiple Domain Controller Options

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
        host="172.16.81.100" port="${jboss.domain.master.port:9990}"/>
      <static-discovery name="backup" protocol="${jboss.domain.master.protocol:remote}"
        host="172.16.81.101" port="${jboss.domain.master.port:9990}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

A static discovery option includes the following required attributes:

name

The name for this domain controller discovery option.

host

The remote domain controller's host name.

port

The remote domain controller's port.

In the example above, the first discovery option is the one expected to succeed. The second can be used in failover situations.

8.5.2. Start a Host Controller with a Cached Domain Configuration

A host controller can be started without a connection to the domain controller by using the **--cached-dc** option; however, the host controller must have previously cached its domain configuration locally from the domain controller. Starting a host controller with this **--cached-dc** option will cache the host controller's domain configuration from the domain controller.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --cached-dc
```

This creates a **domain.cached-remote.xml** file in the **EAP_HOME/domain/configuration/** directory that contains the information necessary for this host controller to temporarily manage its current servers without a domain controller connection.



NOTE

By default, using the **--cached-dc** option only caches configuration used by this host controller, which means that it cannot be promoted to domain controller for the entire domain. See [Cache the Domain Configuration](#) for information on caching the entire domain configuration to allow a host controller to act as the domain controller.

If the domain controller is unavailable when starting this host controller with **--cached-dc**, the host controller will start using the cached configuration saved in the **domain.cached-remote.xml** file. Note that this file must exist or the host controller will fail to start.

While in this state, the host controller cannot modify the domain configuration, but can launch servers and manage deployments.

Once started with the cached configuration, the host controller will continue to attempt to reconnect to the domain controller. Once the domain controller becomes available, the host controller will automatically reconnect to it and synchronize the domain configuration. Note that some configuration changes may require you to reload the host controller to take effect. A warning will be logged on the host controller if this occurs.

8.5.3. Promote a Host Controller to Act as Domain Controller

You can promote a host controller to act as the domain controller if a problem arises with the primary domain controller. The host controller must first [cache the domain configuration locally](#) from the domain controller before it can be [promoted](#).

Cache the Domain Configuration

Use the **--backup** option for any host controller that you might want to become the domain controller.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --backup
```

This creates a **domain.cached-remote.xml** file in the **EAP_HOME/domain/configuration/** directory that contains a copy of the entire domain configuration. This configuration will be used if the host controller is reconfigured to act as the domain controller.

NOTE

The **ignore-unused-configuration** attribute is used to determine how much configuration to cache for a particular host. A value of **true** means that only the configuration relevant to this host controller is cached, which would not allow it to take over as domain controller. A value of **false** means that the entire domain configuration is cached.

The **--backup** argument defaults this attribute to **false** to cache the entire domain. However, if you set this attribute in the **host.xml** file, that value is used.

You can also use the **--cached-dc** option alone to create a copy of the domain configuration, but must set **ignore-unused-configuration** to **false** in the **host.xml** to cache the entire domain. For example:

```
<domain-controller>
  <remote username="$local" security-realm="ManagementRealm" ignore-unused-
configuration="false">
    <discovery-options>
      ...
    </discovery-options>
  </remote>
</domain-controller>
```

Promote a Host Controller to Be the Domain Controller

1. Ensure the original domain controller is stopped.

2. Use the management CLI to connect to the host controller that is to become the new domain controller.
3. Execute the following command to configure the host controller to act as the new domain controller.

```
/host=HOST_NAME:write-local-domain-controller
```

4. Execute the following command to reload the host controller.

```
reload --host=HOST_NAME
```

This host controller will now act as the domain controller.

8.6. MANAGED DOMAIN SETUPS

8.6.1. Set Up a Managed Domain on a Single Machine

You can run multiple host controllers on a single machine by using the **jboss.domain.base.dir** property.



IMPORTANT

It is not supported to configure more than one JBoss EAP host controller as a system service on a single machine.

1. Copy the **EAP_HOME/domain** directory for the domain controller.

```
$ cp -r EAP_HOME/domain /path/to/domain1
```

2. Copy the **EAP_HOME/domain** directory for a host controller.

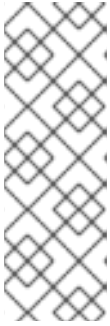
```
$ cp -r EAP_HOME/domain /path/to/host1
```

3. Start the domain controller using **/path/to/domain1**.

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml -  
Djboss.domain.base.dir=/path/to/domain1
```

4. Start the host controller using **/path/to/host1**.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -  
Djboss.domain.base.dir=/path/to/host1 -Djboss.domain.master.address=IP_ADDRESS -  
Djboss.management.http.port=PORT
```

NOTE

When starting a host controller, you must specify the address of the domain controller using the **jboss.domain.master.address** property.

Additionally, since this host controller is running on the same machine as the domain controller, you must change the management interface so that it does not conflict with the domain controller's management interface. This command sets the **jboss.management.http.port** property.

Each instance started in this manner will share the rest of the resources in the base installation directory, for example, **EAP_HOME/modules/**, but use the domain configuration from the directory specified by **jboss.domain.base.dir**.

8.6.2. Set Up a Managed Domain on Two Machines



NOTE

You may need to configure your firewall to run this example.

You can create managed domain on two machines, where one machine is a domain controller and the other machine is a host. For more information, see [About the Domain Controller](#).

- **IP1** = IP address of the domain controller (Machine 1)
- **IP2** = IP address of the host (Machine 2)

Create a Managed Domain on Two Machines

1. On Machine 1

- a. Add a management user so that the host can be authenticated by the domain controller. Use the **add-user.sh** script to add the management user for the host controller, **HOST_NAME**. Make sure to answer **yes** to the last prompt and note the secret value provided. This secret value will be used in the host controller configuration, and will appear similar to the line below:

```
<secret value="SECRET_VALUE" />
```

- b. Start the domain controller.
Specify the **host-master.xml** configuration file, which is preconfigured for a dedicated domain controller. Also, set the **jboss.bind.address.management** property to make the domain controller visible to other machines.

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml -  
Djboss.bind.address.management=IP1
```

2. On Machine 2

- a. Update the host configuration with the user credentials.
Edit **EAP_HOME/domain/configuration/host-slave.xml** and set the host name, **HOST_NAME**, and secret value, **SECRET_VALUE**.

```
<host xmlns="urn:jboss:domain:8.0" name="HOST_NAME">
```

```
<management>
  <security-realms>
    <security-realm name="ManagementRealm">
      <server-identities>
        <secret value="SECRET_VALUE" />
      </server-identities>
    </security-realm>
  </security-realms>
  ...
</management>
```

- b. Start the host controller.

Specify the **host-slave.xml** configuration file, which is preconfigured for a slave host controller. Also, set the **jboss.domain.master.address** property to connect to the domain controller and the **jboss.bind.address** property to set the host controller bind address.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
  Djboss.domain.master.address=IP1 -Djboss.bind.address=IP2
```

You can now manage the domain from the management CLI by specifying the domain controller address with the **--controller** parameter when launching.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --controller=IP1
```

Or you can manage the domain from the management console at **http://IP1:9990**.

8.7. MANAGING MULTIPLE JBOSS EAP VERSIONS

The latest version of JBoss EAP can manage JBoss EAP servers and host that are running an earlier version. See the appropriate section depending on which versions of JBoss EAP you need to manage.

- [Configure a JBoss EAP 7.x Domain Controller to Administer JBoss EAP 6 Instances](#)
- [Configure a JBoss EAP 7.3 domain controller to administer previous minor releases of JBoss EAP](#)

8.7.1. Configure a JBoss EAP 7.x Domain Controller to Administer JBoss EAP 6 Instances

A JBoss EAP 7.1 domain controller can manage hosts and servers running JBoss EAP 6 as long as they are JBoss EAP 6.2 or later.



NOTE

For a JBoss EAP 7.0 domain controller managing JBoss EAP 7.0 hosts that are on a different patch release, there are no configuration changes needed. However, the JBoss EAP 7.0 domain controller must be running a patch release that is equal to or higher than the versions on the host controllers that it manages.

Complete the following tasks to successfully manage JBoss EAP 6 instances in a JBoss EAP 7 managed domain.

1. [Add the JBoss EAP 6 configuration to the JBoss EAP 7 domain controller](#) .
2. [Update the behavior for the JBoss EAP 6 profiles](#) .

3. [Set the server group for the JBoss EAP 6 servers](#) .
4. [Prevent the JBoss EAP 6 instances from receiving JBoss EAP 7 updates](#) .

Once these tasks are complete, you can manage your JBoss EAP 6 servers and configurations from the JBoss EAP 7 domain controller using the management CLI. Note that JBoss EAP 6 hosts will not be able to take advantage of new JBoss EAP 7 features, such as batch processing.



WARNING

Because the management console is optimized for the latest version of JBoss EAP, you should not use it to update your JBoss EAP 6 hosts, servers, and profiles. Use the management CLI instead when managing your JBoss EAP 6 configurations from a JBoss EAP 7 managed domain.

8.7.1.1. Add the JBoss EAP 6 Configuration to the JBoss EAP 7 Domain Controller

To allow the domain controller to manage your JBoss EAP 6 servers, you must provide the JBoss EAP 6 configuration details in the JBoss EAP 7 domain configuration. You can do this by copying the JBoss EAP 6 profiles, socket binding groups, and server groups to the JBoss EAP 7 **domain.xml** configuration file.

You will need to rename resources if any conflict with the existing names in the JBoss EAP 7 configuration. There are also some additional [adjustments](#) to make to ensure the proper behavior.

The following procedure uses the JBoss EAP 6 **default** profile, **standard-sockets** socket binding group, and **main-server-group** server group.

1. Edit the JBoss EAP 7 **domain.xml** configuration file. It is recommended to back up this file before editing.
2. Copy the applicable JBoss EAP 6 profiles to the JBoss EAP 7 **domain.xml** file.
This procedure assumes that the JBoss EAP 6 **default** profile was copied and renamed to **eap6-default**.

JBoss EAP 7 domain.xml

```
<profiles>
...
<profile name="eap6-default">
...
</profile>
</profiles>
```

3. Add the necessary extensions used by this profile.
If your JBoss EAP 6 profile uses subsystems that are no longer present in JBoss EAP 7, you must add the appropriate extensions to the JBoss EAP domain configuration.

JBoss EAP 7 domain.xml

```
<extensions>
...
<extension module="org.jboss.as.configadmin"/>
<extension module="org.jboss.as.threads"/>
<extension module="org.jboss.as.web"/>
</extensions>
```

- Copy the applicable JBoss EAP 6 socket binding groups to the JBoss EAP 7 **domain.xml** file. This procedure assumes that the JBoss EAP 6 **standard-sockets** socket binding group was copied and renamed to **eap6-standard-sockets**.

JBoss EAP 7 domain.xml

```
<socket-binding-groups>
...
<socket-binding-group name="eap6-standard-sockets" default-interface="public">
...
</socket-binding-group>
</socket-binding-groups>
```

- Copy the applicable JBoss EAP 6 server groups to the JBoss EAP 7 **domain.xml** file. This procedure assumes that the JBoss EAP 6 **main-server-group** server group was copied and renamed to **eap6-main-server-group**. You must also update this server group to use the JBoss EAP 6 profile, **eap6-default**, and the JBoss EAP 6 socket binding group, **eap6-standard-sockets**.

JBoss EAP 7 domain.xml

```
<server-groups>
...
<server-group name="eap6-main-server-group" profile="eap6-default">
...
  <socket-binding-group ref="eap6-standard-sockets"/>
</server-group>
</server-groups>
```

8.7.1.2. Update the Behavior for the JBoss EAP 6 Profiles

Additional updates to the profiles used by your JBoss EAP 6 instances are necessary depending on the JBoss EAP version and desired behavior. You may require additional changes depending on the subsystems and configuration that your existing JBoss EAP 6 instances use.

Start the JBoss EAP 7 domain controller and launch its management CLI to perform the following updates. These examples assume that the JBoss EAP 6 profile is **eap6-default**.

- Remove the **bean-validation** subsystem.
JBoss EAP 7 moved bean validation functionality from the **ee** subsystem into its own subsystem, **bean-validation**. If a JBoss EAP 7 domain controller sees a legacy **ee** subsystem, it adds the new **bean-validation** subsystem. However, the JBoss EAP 6 hosts will not recognize this subsystem, so it must be removed.

JBoss EAP 7 Domain Controller CLI

```
/profile=eap6-default/subsystem=bean-validation:remove
```

-
- Set CDI 1.0 behavior.

This is only necessary if you want CDI 1.0 behavior for your JBoss EAP 6 servers, as opposed to behavior of later CDI versions used in JBoss EAP 7. If you want CDI 1.0 behavior, make the following updates to the **weld** subsystem.

JBoss EAP 7 Domain Controller CLI

```
/profile=eap6-default/subsystem=weld:write-attribute(name=require-bean-descriptor,value=true)
```

```
/profile=eap6-default/subsystem=weld:write-attribute(name=non-portable-mode,value=true)
```

- Enable datasource statistics for JBoss EAP 6.2.

This is only necessary if your profile is being used by JBoss EAP 6.2 servers. JBoss EAP 6.3 introduced the **statistics-enabled** attribute, which defaults to **false** to not collect statistics; however, the JBoss EAP 6.2 behavior was to collect statistics. If this profile is used by a JBoss EAP 6.2 host and a host running a newer JBoss EAP version, the behavior would be inconsistent between hosts, which is not allowed. Therefore, profiles intended for use by a JBoss EAP 6.2 host should make the following change for their datasources.

JBoss EAP 7 Domain Controller CLI

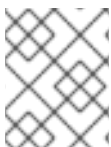
```
/profile=eap6-default/subsystem=datasources/data-source=ExampleDS:write-attribute(name=statistics-enabled,value=true)
```

8.7.1.3. Set the Server Group for the JBoss EAP 6 Servers

If you renamed the server groups, you need to update the JBoss EAP 6 host configuration to use the new server groups specified in the JBoss EAP 7 configuration. This example uses the **eap6-main-server-group** server group specified in the JBoss EAP 7 **domain.xml**.

JBoss EAP 6 host-slave.xml

```
<servers>
  <server name="server-one" group="eap6-main-server-group"/>
  <server name="server-two" group="eap6-main-server-group">
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```



NOTE

A host cannot use features or configuration settings that were introduced in a newer version of JBoss EAP than the one the host is running.

8.7.1.4. Prevent the JBoss EAP 6 Instances from Receiving JBoss EAP 7 Updates

The domain controller in a managed domain forwards configuration updates to its host controllers. You must use the **host-exclude** configuration to specify the resources that should be hidden from particular versions. Choose the appropriate preconfigured **host-exclude** option for your JBoss EAP 6 version: **EAP62**, **EAP63**, **EAP64**, or **EAP64z**.

The **active-server-groups** attribute of the **host-exclude** configuration specifies the list of server groups that are used by a particular version. These server groups and their associated profiles, socket binding groups, and deployment resources will be available to hosts of this version, but all others will be hidden from these hosts.

This example assumes that the version is JBoss EAP 6.4.z and adds the JBoss EAP 6 server group **eap6-main-server-group** as an active server group.

JBoss EAP 7 Domain Controller CLI

```
/host-exclude=EAP64z:write-attribute(name=active-server-groups,value=[eap6-main-server-group])
```

If necessary, you can specify additional socket binding groups used by your servers using the **active-socket-binding-groups** attribute. This is only required for socket binding groups that are not associated with the server groups specified in **active-server-groups**.

8.7.2. Configure a JBoss EAP 7.3 Domain Controller to Administer Previous Minor Releases of JBoss EAP

A JBoss EAP 7.3 domain controller can manage hosts and servers running from a previous minor release of JBoss EAP.



NOTE

For a JBoss EAP 7.3 domain controller managing JBoss EAP 7.2 hosts that are on a different patch release, you do not need to make any configuration changes. However, you must run the JBoss EAP 7.2 domain controller on a patch release that is equal to or higher than the versions on the host controllers that it manages.

Complete the following tasks to successfully manage JBoss EAP 7.2 instances in a JBoss EAP 7.3 managed domain.

1. [Add the JBoss EAP 7.2 configuration to the JBoss EAP 7.3 domain controller](#) .
2. [Set the server group for the JBoss EAP 7.2 servers](#).
3. [Prevent the JBoss EAP 7.2 instances from receiving JBoss EAP 7.3 updates](#) .

After you complete these tasks, you can manage your JBoss EAP 7.2 servers and configurations from the JBoss EAP 7.3 domain controller using the management CLI.



WARNING

Because the management console is optimized for the latest version of JBoss EAP, you must use the CLI to manage your JBoss EAP 7.2 configurations from a JBoss EAP 7.3 managed domain. Do not use the management console to update JBoss EAP 7.2 hosts, servers, and profiles.

8.7.2.1. Add the JBoss EAP 7.2 Configuration to the JBoss EAP 7.3 Domain Controller

To enable the domain controller to manage your JBoss EAP 7.2 servers, you must provide the JBoss EAP 7.2 configuration details in the JBoss EAP 7.3 domain configuration. You can do this by copying the JBoss EAP 7.2 profiles, socket binding groups, and server groups to the JBoss EAP 7.3 **domain.xml** configuration file.

You must rename a resource if its name conflicts with resource names in the JBoss EAP 7.3 configuration.

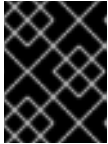
The following procedure uses the JBoss EAP 7.2 **default** profile, **standard-sockets** socket binding group, and **main-server-group** server group.

Prerequisites

- You have copied and renamed the JBoss EAP 7.2 **default** profile to **eap72-default**.
- You have copied and renamed the JBoss EAP 7.2 **standard-sockets** socket binding group to **eap72-standard-sockets**.
- You have copied and renamed the JBoss EAP 7.2 **main-server-group** server group to **eap72-main-server-group**.
 - You have updated the server group to use the JBoss EAP 7.2 profile, **eap72-default**, and to use the JBoss EAP 7.2 socket binding group, **eap72-standard-sockets**.

Procedure

1. Edit the JBoss EAP 7.3 **domain.xml** configuration file.



IMPORTANT

Back up the JBoss EAP 7.3 **domain.xml** configuration file before you edit the file.

2. Copy the applicable JBoss EAP 7.2 profiles to the JBoss EAP 7.3 **domain.xml** file. For example:

```
<profiles>
...
<profile name="eap72-default">
...
</profile>
</profiles>
```

3. Copy the applicable JBoss EAP 7.2 socket binding groups to the JBoss EAP 7.3 **domain.xml** file. For example:

```
<socket-binding-groups>
...
<socket-binding-group name="eap72-standard-sockets" default-interface="public">
...
</socket-binding-group>
</socket-binding-groups>
```

4. Copy the applicable JBoss EAP 7.2 server groups to the JBoss EAP 7.3 **domain.xml** file. For example:

■

```

<server-groups>
...
<server-group name="eap72-main-server-group" profile="eap72-default">
...
  <socket-binding-group ref="eap72-standard-sockets"/>
</server-group>
</server-groups>

```

8.7.2.2. Set the Server Group for the JBoss EAP7.2 Servers

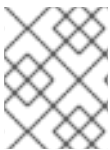
If you renamed the server groups, you need to update the JBoss EAP 7.2 host configuration to use the new server groups specified in the JBoss EAP 7.3 configuration. This example uses the **eap72-main-server-group** server group specified in the JBoss EAP 7.3 **domain.xml**.

JBoss EAP 7.2 host-slave.xml

```

<servers>
  <server name="server-one" group="eap72-main-server-group"/>
  <server name="server-two" group="eap72-main-server-group">
    <socket-bindings port-offset="150"/>
  </server>
</servers>

```



NOTE

A host cannot use features or configuration settings that were introduced in a newer version of JBoss EAP than the one the host is running.

8.7.2.3. Prevent the JBoss EAP 7.2 Instances from Receiving JBoss EAP 7.3 Updates

A managed domain controller forwards configuration updates to its host controllers, so that a JBoss EAP 7.2 host does not receive configuration and resources that are specific to JBoss EAP 7.3. You must configure the JBoss EAP 7.2 host to ignore those resources. You can do this by setting the **ignore-unused-configuration** attribute on the JBoss EAP 7.2 host.



NOTE

You can also use the **host-exclude** configuration to instruct the domain controller to hide certain resources from hosts running certain JBoss EAP versions. For an example of how to use the **host-exclude** configuration, see [Prevent the JBoss EAP 6 Instances from Receiving JBoss EAP 7 Updates](#). For JBoss EAP 7.2, you use the **EAP72 host-exclude** option.

Set the **ignore-unused-configuration** attribute to **true** in the JBoss EAP 7.2 host controller connection configuration to the remote domain controller.

JBoss EAP 7.2 host-slave.xml

```

<domain-controller>
  <remote security-realm="ManagementRealm" ignore-unused-configuration="true">
    <discovery-options>
      <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
        host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9990}"/>
    </discovery-options>
  </remote>
</domain-controller>

```



```

</discovery-options>
</remote>
</domain-controller>

```

With this setting, only the server groups used by this host, and their associated profiles, socket binding groups, and deployment resources, are made available to the host. All other resources are ignored.

8.8. MANAGING JBOSS EAP PROFILES

8.8.1. About Profiles

JBoss EAP uses *profiles* as a way to organize which subsystems are available to a server. A profile consists of a collection of available subsystems along with each subsystem's specific configuration. A profile with a large number of subsystems results in a server with a large set of capabilities. A profile with a small, focused set of subsystems will have fewer capabilities but a smaller footprint.

JBoss EAP comes with five predefined profiles that should satisfy most use cases:

default

Includes commonly used subsystems, such as **logging**, **security**, **datasources**, **infinispan**, **webservices**, **ee**, **ejb3**, **transactions**, and so on.

ha

Includes the subsystems provided in the *default* profile with the addition of the **jgroups** and **modcluster** subsystems for high availability

full

Includes the subsystems provided in the *default* profile with the addition of the **messaging-activemq** and **iiop-openjdk** subsystems

full-ha

Includes the subsystems provided in the *full* profile with the addition of the **jgroups** and **modcluster** subsystems for high availability

load-balancer

Includes the minimum subsystems necessary to use the built-in mod_cluster front-end load balancer to load balance other JBoss EAP instances.



NOTE

JBoss EAP offers the ability to disable extensions or unload drivers and other services manually by removing the subsystems from the configuration of existing profiles. However, for most cases this is unnecessary. Since JBoss EAP dynamically loads subsystems as they are needed, if the server or an application never use a subsystem, it will not be loaded.

In cases where the existing profiles do not provide the necessary capabilities, JBoss EAP also provides the ability to define custom profiles as well.

8.8.2. Cloning a Profile

JBoss EAP allows you to create a new profile in a managed domain by cloning an existing profile. This will create a copy of the original profile's configuration and subsystems.

A profile can be cloned using the management CLI by using the **clone** operation on the desired profile to clone.

```
/profile=full-ha:clone(to-profile=cloned-profile)
```

You can also clone a profile from the management console by selecting the desired profile to clone and clicking **Clone**.

8.8.3. Creating Hierarchical Profiles

In a managed domain, you can create a hierarchy of profiles. This allows you to create a base profile with common extensions that other profiles can inherit.

The managed domain defines several profiles in **domain.xml**. If multiple profiles use the same configuration for a particular subsystem, you can configure it in just one place instead of different profiles. The values in parent profiles cannot be overridden.

In addition, each profile must be self-sufficient. If an element or subsystem is referenced, then it must be defined in the profile where it is referenced.

A profile can include other profiles in a hierarchy using the management CLI by using the **list-add** operation and providing the profile to include.

```
/profile=new-profile:list-add(name=includes, value=PROFILE_NAME)
```

CHAPTER 9. CONFIGURING JVM SETTINGS

Configuration of Java Virtual Machine (JVM) settings is different for a standalone JBoss EAP server, or a JBoss EAP server in a managed domain.

For a standalone JBoss EAP server instance, the server startup processes pass JVM settings to the JBoss EAP server at startup. These can be declared from the command line before launching JBoss EAP, or using the **System Properties** page under **Configuration** in the management console.

In a managed domain, JVM settings are declared in the **host.xml** and **domain.xml** configuration files, and can be configured at host, server group, or server levels.



NOTE

System properties must be configured in **JAVA_OPTS** to be used by JBoss EAP modules (such as the logging manager) during startup.

9.1. CONFIGURING JVM SETTINGS FOR A STANDALONE SERVER

JVM settings for standalone JBoss EAP server instances can be declared at runtime by setting the **JAVA_OPTS** environment variable before starting the server.

An example of setting the **JAVA_OPTS** environment variable on Linux is shown below.

```
$ export JAVA_OPTS="-Xmx1024M"
```

The same setting can be used in a Microsoft Windows environment:

```
set JAVA_OPTS="Xmx1024M"
```

Alternatively, JVM settings can be added to the **standalone.conf** file, or **standalone.conf.bat** for Windows Server, in the **EAP_HOME/bin** folder, which contains examples of options to pass to the JVM.



WARNING

Setting the **JAVA_OPTS** environment variable will override the default values from **standalone.conf**, which may cause JBoss EAP startup issues.

9.2. CONFIGURING JVM SETTINGS FOR A MANAGED DOMAIN

In a JBoss EAP managed domain, you can define JVM settings at multiple levels. You can define custom JVM settings on a particular host, and then apply those settings to server groups, or to individual server instances.

By default, server groups and individual servers will inherit the JVM settings from their parent, but you can choose to override JVM settings at each level.

**NOTE**

The JVM settings in **domain.conf**, or **domain.conf.bat** for Windows Server, are applied to the Java process of the JBoss EAP host controller, and not the individual JBoss EAP server instances controlled by that host controller.

9.2.1. Defining JVM Settings on a Host Controller

You can define JVM settings on a host controller, and apply those settings to server groups or individual servers. JBoss EAP comes with a **default** JVM setting, but the following management CLI command demonstrates creating a new JVM setting named **production_jvm** with some custom JVM settings and options.

```
/host=HOST_NAME/jvm=production_jvm:add(heap-size=2048m, max-heap-size=2048m, max-permgen-size=512m, stack-size=1024k, jvm-options=["-XX:-UseParallelGC"])
```

See [Managed Domain JVM Configuration Attributes](#) for descriptions of all available options.

You can also create and edit JVM settings in the JBoss EAP management console by navigating to **Runtime → Hosts**, choosing a host and clicking **View**, and selecting the **JVMs** tab.

These settings are stored in the within the **<jvm>** tag in **host.xml**.

9.2.2. Applying JVM Settings to a Server Group

When creating a server group, you can specify a JVM configuration that all servers in the group will use. The following management CLI commands demonstrate creating a server group name **groupA** that uses the **production_jvm** JVM settings that were shown in [the previous example](#).

```
/server-group=groupA:add(profile=default, socket-binding-group=standard-sockets)
/server-group=groupA/jvm=production_jvm:add
```

All servers in the server group will inherit JVM settings from **production_jvm**.

You can also override specific JVM settings at the server group level. For example, to set a different heap size, you can use the following command:

```
/server-group=groupA/jvm=production_jvm:write-attribute(name=heap-size,value="1024m")
```

After applying the above command, the server group **groupA** will inherit the JVM settings from **production_jvm**, except for the heap size which has an overridden value of **1024m**.

See [Managed Domain JVM Configuration Attributes](#) for descriptions of all available options.

You can also edit server group JVM settings in the JBoss EAP management console by navigating to **Runtime → Server Groups**, choosing a server group and clicking **View**, and selecting the **JVMs** tab.

These settings for a server group are stored in **domain.xml**.

9.2.3. Applying JVM Settings to an Individual Server

By default, an individual JBoss EAP server instance will inherit the JVM settings of the server group it belongs to. However, you can choose to override the inherited settings with another complete JVM setting definition from the host controller, or choose to override specific JVM settings.

For example, the following command overrides the JVM definition of the [server group in the previous example](#), and sets the JVM settings for **server-one** to the **default** JVM definition:

```
/host=HOST_NAME/server-config=server-one/jvm=default:add
```

Also, similar to server groups, you can override specific JVM settings at the server level. For example, to set a different heap size, you can use the following command:

```
/host=HOST_NAME/server-config=server-one/jvm=default:write-attribute(name=heap-size,value="1024m")
```

See [Managed Domain JVM Configuration Attributes](#) for descriptions of all available options.

You can also edit server JVM settings in the JBoss EAP management console by navigating to **Runtime** → **Hosts**, choosing the host, clicking **View** on the server, and selecting the **JVMs** tab.

These settings for an individual server are stored in **host.xml**.

9.3. DISPLAYING THE JVM STATUS

You can view the status of JVM resources, such as heap and thread usage, for standalone or managed domain servers from the management console. While statistics are not displayed in real time, you can click **Refresh** to provide an up-to-date overview of JVM resources.

To display the JVM status for a standalone JBoss EAP server:

- Navigate to the **Runtime** tab, select the server, and select **Status**.

To display the JVM status for a JBoss EAP server in a managed domain:

- Navigate to **Runtime** → **Hosts**, select the host and server, and select **Status**.

This shows the following heap usage information:

Max

The maximum amount of memory that can be used for memory management.

Used

The amount of used memory.

Committed

The amount of memory that is committed for the Java Virtual Machine to use.

Other information, such as JVM uptime and thread usage, is also available.

9.4. TUNING THE JVM

For tips on optimizing JVM performance, see the [JVM Tuning](#) section of the *Performance Tuning Guide*.

CHAPTER 10. MAIL SUBSYSTEM

10.1. CONFIGURING THE MAIL SUBSYSTEM

The **mail** subsystem allows you to configure mail sessions in JBoss EAP and then inject those sessions into applications using JNDI. It also supports configuration using the Jakarta EE **@MailSessionDefinition** and **@MailSessionDefinitions** annotations.

Configuring SMTP server for use in an application

1. Configure the SMTP server and the outbound socket binding using the following CLI commands, for example:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp:add(host=localhost, port=25)
```

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp, username=user, password=pass, tls=true)
```

2. Call the configured mail session within an application

```
@Resource(lookup="java:jboss/mail/MySession")
private Session session;
```

For a full list of the attributes available for configuring mail sessions and servers, see [Mail Subsystem Attributes](#).

10.2. CONFIGURING CUSTOM TRANSPORTS

When using a standard mail server, such as POP3 or IMAP, the mail server has a set of attributes that can be defined. Some of these attributes are mandatory. The most important of these is the **outbound-socket-binding-ref**, which is a reference to the outbound mail socket binding and is defined with a host address and port number.

Defining the **outbound-socket-binding-ref** may not be the most effective solution for users who have their host configuration using multiple hosts for load balancing purposes. Standard Jakarta Mail does not support host configuration using multiple hosts for load balancing. Therefore, users who have this configuration using multiple hosts are required to implement custom mail transports. These custom mail transports do not require the **outbound-socket-binding-ref** and allow custom host property formats.

You can configure custom mail transports from the management CLI.

1. Add a new mail session and specify the JNDI name.

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

2. Add an outbound socket binding and specify the host and port.

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp-binding:add(host=localhost, port=25)
```

3. Add an SMTP server and specify the outbound socket binding, username, and password.

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-
ref=my-smtp-binding, username=user, password=pass, tls=true)
```

NOTE

You can configure a POP3 or IMAP server using similar steps.

POP3 Server

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=my-pop3-binding:add(host=localhost, port=110)
/subsystem=mail/mail-session=mySession/server=pop3:add(outbound-socket-binding-
ref=my-pop3-binding, username=user, password=pass)
```

IMAP Server

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=my-imap-binding:add(host=localhost, port=143)
/subsystem=mail/mail-session=mySession/server=imap:add(outbound-socket-binding-
ref=my-imap-binding, username=user, password=pass)
```

To use a custom server, create a custom mail server without an outbound socket binding. You can specify the host information in the properties definition of the custom mail server. For example:

```
/subsystem=mail/mail-
session=mySession/custom=myCustomServer:add(username=user,password=pass, properties=
{"host" => "myhost", "my-property" => "value"})
```

If you define a custom protocol, any property name that includes a period (.) is considered a fully-qualified name and is passed directly. Any other format, such as **my-property**, is translated in the following format: **mail.server-name.my-property**.

The following XML is an example mail configuration that includes custom servers.

```
<subsystem xmlns="urn:jboss:domain:mail:3.0">
  <mail-session name="default" jndi-name="java:jboss/mail/Default">
    <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
  <mail-session name="myMail" from="user.name@domain.org" jndi-name="java:Mail">
    <smtp-server password="password" username="user" tls="true" outbound-socket-binding-
ref="mail-smtp"/>
    <pop3-server outbound-socket-binding-ref="mail-pop3"/>
    <imap-server password="password" username="nobody" outbound-socket-binding-ref="mail-
imap"/>
  </mail-session>
  <mail-session name="custom" jndi-name="java:jboss/mail/Custom" debug="true">
    <custom-server name="smtp" password="password" username="username">
      <property name="host" value="mail.example.com"/>
    </custom-server>
  </mail-session>
  <mail-session name="custom2" jndi-name="java:jboss/mail/Custom2" debug="true">
```

```

<custom-server name="pop3" outbound-socket-binding-ref="mail-pop3">
  <property name="custom-prop" value="some-custom-prop-value"/>
</custom-server>
</mail-session>
</subsystem>

```

10.3. USE A CREDENTIAL STORE FOR PASSWORDS

In addition to providing clear-text passwords in the **mail** subsystem, you can also use a credential store to provide passwords. The **elytron** subsystem provides the ability to create credential stores to securely house and use your passwords throughout JBoss EAP. You can find more details on creating and using credential stores in the [Credential Store](#) section of *How to Configure Server Security*.

Using a Credential Store Using the Management CLI

```

/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp-
binding, username=user, credential-reference={store=exampleCS, alias=mail-session-pw}, tls=true)

```



NOTE

The following is an example of how to specify a **credential-reference** attribute that uses a clear-text password.

```
credential-reference={clear-text="MASK-Ewcyuqd/nP9;A1B2C3D4;351"}
```

Using a Credential Store Using the Management Console

1. Access the management console. For more information, see the [Management Console](#).
2. Navigate to **Configuration** → **Subsystems** → **Mail**.
3. Select the appropriate mail session and click **View**.
4. Select **Server** and choose the appropriate mail session server. You can configure the credential reference in the **Credential Reference** tab and edit other attributes in the **Attributes** tab.

CHAPTER 11. LOGGING WITH JBOSS EAP

JBoss EAP provides highly-configurable logging facilities for both its own internal use and for use by deployed applications. The **logging** subsystem is based on JBoss LogManager and supports several third-party application logging frameworks in addition to JBoss Logging.

11.1. ABOUT SERVER LOGGING

11.1.1. Server Logging

By default, all JBoss EAP log entries are written to the **server.log** file. The location of this file depends on your operating mode.

- Standalone server: ***EAP_HOME/standalone/log/server.log***
- Managed domain: ***EAP_HOME/domain/servers/SERVER_NAME/log/server.log***

This file is often referred to as the server log. For more information, see the [Root Logger](#) section.

11.1.2. Bootup Logging

During bootup, JBoss EAP logs information about the Java environment and the startup of each service. This log can be useful when troubleshooting. By default, all log entries are written to the [server log](#).

Bootup logging configuration is specified in the **logging.properties** configuration file, which is active until the JBoss EAP **logging** subsystem is started and takes over. The location of this file depends on your operating mode.

- Standalone server: ***EAP_HOME/standalone/configuration/logging.properties***
- Managed domain:
 - There is a **logging.properties** file for the domain controller and for each server.
 - Domain controller: ***EAP_HOME/domain/configuration/logging.properties***
 - Server: ***EAP_HOME/domain/servers/SERVER_NAME/data/logging.properties***



WARNING

It is recommended that you do not directly edit the **logging.properties** file unless you know of a specific use case that requires it. Before doing so, it is recommended that you open a support case from the [Red Hat Customer Portal](#).

Changes made manually to the **logging.properties** file are overwritten on startup.

11.1.2.1. View Bootup Errors

When troubleshooting JBoss EAP, checking for errors that occurred during bootup should be one of the first steps taken. You can then use the information provided to diagnose and resolve their causes. Open a support case for assistance in troubleshooting bootup errors.

There are two methods of viewing bootup errors, each with its advantages. You can examine the [server.log](#) file or read the boot errors using the [read-boot-errors](#) management CLI command.

Examine the Server Log File

You can open the **server.log** file to view any errors that occurred during bootup.

This method allows you to see each error message together with possibly related messages, allowing you to get more information about *why* an error might have occurred. It also allows you to see error messages in plain text format.

1. Open the file **server.log** in a file viewer.
2. Navigate to the end of the file.
3. Search backward for the **WFLYSRV0049** message identifier, which marks the start of the latest bootup sequence.
4. Search the log from that point onward for instances of **ERROR**. Each instance will include a description of the error and list the modules involved.

The following is an example error description from the **server.log** log file.

```
2016-03-16 14:32:01,627 ERROR [org.jboss.msc.service.fail] (MSC service thread 1-7) MSC000001:
Failed to start service jboss.undertow.listener.default: org.jboss.msc.service.StartException in service
jboss.undertow.listener.default: Could not start http listener
    at org.wildfly.extension.undertow.ListenerService.start(ListenerService.java:142)
    at
org.jboss.msc.service.ServiceControllerImpl$StartTask.startService(ServiceControllerImpl.java:1948)
    at org.jboss.msc.service.ServiceControllerImpl$StartTask.run(ServiceControllerImpl.java:1881)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
    at java.lang.Thread.run(Thread.java:745)
Caused by: java.net.BindException: Address already in use
...
```

Read the Boot Errors from the Management CLI

You can use the **read-boot-errors** management CLI command to view errors if a server starts but reported errors during bootup.

This method does not require access to the server's file system, which is useful for anyone responsible for monitoring for errors who does not have file system access. Since it is a management CLI command, it can be used in a script. For example, you could write a script that starts multiple JBoss EAP instances, then checks for errors that occurred on bootup.

Run the following management CLI command.

```
/core-service=management:read-boot-errors
```

Any errors that occurred during bootup will be listed.

```
{
  "outcome" => "success",
```

```

"result" => [
  {
    "failed-operation" => {
      "operation" => "add",
      "address" => [
        ("subsystem" => "undertow"),
        ("server" => "default-server"),
        ("http-listener" => "default")
      ]
    },
    "failure-description" => "{\"WFLYCTL0080: Failed services\" =>
    {\"jboss.undertow.listener.default\" => \"org.jboss.msc.service.StartException in service
    jboss.undertow.listener.default: Could not start http listener
    Caused by: java.net.BindException: Address already in use\"}},
    \"failed-services\" => {\"jboss.undertow.listener.default\" =>
    \"org.jboss.msc.service.StartException in service jboss.undertow.listener.default: Could not start http
    listener
    Caused by: java.net.BindException: Address already in use\"}
    }
    ...
  ]
}

```

11.1.3. Garbage Collection Logging

Garbage collection logging logs all garbage collection activity to plain text log files. These log files can be useful for diagnostic purposes. Garbage collection logging is enabled by default for a JBoss EAP standalone server on all supported configurations *except* IBM Java development kit.

The location of the garbage collection log is ***EAP_HOME/standalone/log/gc.log.DIGIT.current***. Garbage collection logs are limited to 3 MB each, and up to five files are rotated.

It is strongly recommended to leave garbage collection logging enabled as it can be useful in troubleshooting and should have minimal overhead. However, you can disable garbage collection logging for a standalone server by setting the **GC_LOG** variable to **false** before starting the server. For example:

```

$ export GC_LOG=false
$ EAP_HOME/bin/standalone.sh

```

11.1.4. Default Log File Locations

The following log files are created for the default logging configurations. The default configuration writes the server log files using periodic log handlers.

Table 11.1. Default Log File for a Standalone Server

Log File	Description
<i>EAP_HOME/standalone/log/server.log</i>	Contains server log messages, including server startup messages.
<i>EAP_HOME/standalone/log/gc.log.DIGIT.current</i>	Contains garbage collection details.

Table 11.2. Default Log Files for a Managed Domain

Log File	Description
<code>EAP_HOME/domain/log/host-controller.log</code>	Contains log messages related to the startup of the host controller.
<code>EAP_HOME/domain/log/process-controller.log</code>	Contains log messages related to the startup of the process controller.
<code>EAP_HOME/domain/servers/SERVER_NAME/log/server.log</code>	Contains log messages for the named server, including server startup messages.

11.1.5. Set the Default Locale of the Server

You can configure the default locale for JBoss EAP by setting JVM properties in the appropriate startup configuration file. The startup configuration file is **`EAP_HOME/bin/standalone.conf`** for a standalone server or **`EAP_HOME/bin/domain.conf`** for a managed domain.



NOTE

For Windows Server, the JBoss EAP startup configuration files are **`standalone.conf.bat`** and **`domain.conf.bat`**.

Log messages that have been internationalized and localized will use this default locale. See the JBoss EAP *Development Guide* for information on [creating internationalized log messages](#).

Set the Language

Specify the language by setting the **`user.language`** property using the **`JAVA_OPTS`** variable. For example, add the following line to the startup configuration file to set a French locale.

```
JAVA_OPTS="$JAVA_OPTS -Duser.language=fr"
```

Log messages that have been internationalized and localized will now output in French.

Set the Language and Country

In addition to the language, it may also be necessary to specify the country by setting the **`user.country`** property. For example, add the following line to the startup configuration file to set the Portuguese locale for Brazil.

```
JAVA_OPTS="$JAVA_OPTS -Duser.language=pt -Duser.country=BR"
```

Log messages that have been internationalized and localized will now output in Brazilian Portuguese.

Set the Server Locale Using the `org.jboss.logging.locale` Property

You can configure the **`org.jboss.logging.locale`** property to override the locale for messages logged using JBoss Logging, including any messages from JBoss EAP and its owned dependencies. Other dependencies, such as JSF, cannot get an overridden locale.

To start the JBoss EAP server with a different locale than the system default, you can edit **`EAP_HOME/bin/standalone.conf`** or the **`EAP_HOME/bin/domain.conf`** file, depending on your operating mode, and append the following command to set the JVM parameter for the required locale.

The value of the property must be specified in the BCP 47 format. For example, to set Brazilian Portuguese, use pt-BR.

```
JAVA_OPTS="$JAVA_OPTS -Dorg.jboss.logging.locale=pt-BR"
```

11.2. VIEWING LOG FILES

Viewing server and application logs is important in order to help diagnose errors, performance problems, and other issues. Some users may prefer to view logs directly on the server file system. For those who do not have direct access to the file system, or who prefer a graphical interface, JBoss EAP allows you to view logs from the management console. You can also view logs using the management CLI.

For a log to be accessible from one of the management interfaces, it must be located in the directory specified by the server's **jboss.server.log.dir** property and be defined as a file, periodic rotating, size rotating, or periodic size rotating log handler. RBAC role assignments are also honored, so a user logged in to the management console or CLI can only view logs that they are authorized to access.

View Logs from the Management Console

You can view logs directly from the management console.

1. Select the **Runtime** tab and select the appropriate server.
2. Select **Log Files** and choose a log file from the list.
3. Click **View** to view and search log contents, or choose **Download** from the drop down to download the log file to your local file system.



WARNING

The management console log viewer is not intended to be a text editor replacement for viewing very large log files, for example, greater than 100MB. You will be prompted for confirmation if you attempt to open a log file that is larger than 15MB. Opening a very large file in the management console could crash your browser, so you should always download large log files locally and open them in a text editor.

View Logs from the Management CLI

You can read the contents of log files from the management CLI using the **read-log-file** command. By default, this displays the last **10** lines of the specified log file.

```
/subsystem=logging/log-file=LOG_FILE_NAME:read-log-file
```



NOTE

In a managed domain, precede this command with **/host=HOST_NAME/server=SERVER_NAME**.

You can use the following parameters to customize the log output.

encoding

The character encoding used to read the file.

lines

The number of lines to read from the file. A value of **-1** will read all log lines. The default is **10**.

skip

The number of lines to skip before reading. The default is **0**.

tail

Whether to read from the end of the file. Defaults to **true**.

For example, the following management CLI command reads the first **5** lines from the top of the **server.log** log file.

```
/subsystem=logging/log-file=server.log:read-log-file(lines=5,tail=false)
```

This produces the following output.

```
{
  "outcome" => "success",
  "result" => [
    "2016-03-24 08:49:26,612 INFO [org.jboss.modules] (main) JBoss Modules version 1.5.1.Final-redhat-1",
    "2016-03-24 08:49:26,788 INFO [org.jboss.msc] (main) JBoss MSC version 1.2.6.Final-redhat-1",
    "2016-03-24 08:49:26,863 INFO [org.jboss.as] (MSC service thread 1-7) WFLYSRV0049: JBoss EAP 7.0.0.GA (WildFly Core 2.0.13.Final-redhat-1) starting",
    "2016-03-24 08:49:27,973 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0039: Creating http management service using socket-binding (management-http)",
    "2016-03-24 08:49:27,994 INFO [org.xnio] (MSC service thread 1-1) XNIO version 3.3.4.Final-redhat-1"
  ]
}
```

11.3. ABOUT THE LOGGING SUBSYSTEM

The JBoss EAP **logging** subsystem is configured using a system of [log categories](#) and [log handlers](#). Log categories define what messages to capture. Log handlers define how to deal with those messages, for example, writing to a disk or sending to the console.

[Logging profiles](#) allow uniquely-named sets of logging configuration to be created and assigned to applications independent of any other logging configuration. The configuration of logging profiles is almost identical to the main **logging** subsystem.

11.3.1. Root Logger

The JBoss EAP root logger captures all log messages, of the specified log level or higher, sent to the server that are not captured by a log category.

By default, the root logger is configured to use a console and a periodic log handler. The periodic log handler is configured to write to the **server.log** file. This file is often referred to as the server log.

See [Configuring the Root Logger](#) for more information.

11.3.2. Log Categories

A log category defines a set of log messages to capture and one or more log handlers that will process the messages.

The log messages to capture are defined by the specified Java package of origin and log level. Messages from classes in that package and of that log level or higher are captured by the log category and sent to the specified log handlers.



NOTE

Although the log category is typically the Java package and class name, it can be any name specified by the **Logger.getLogger(*LOGGER_NAME*)** method.

Log categories can optionally use the log handlers of the root logger instead of their own handlers.

See [Configuring Log Categories](#) for more information.

11.3.3. Log Handlers

Log handlers define how captured log messages are recorded. The available log handler types are *console*, *file*, *periodic*, *size*, *periodic size*, *syslog*, *custom*, and *async*.



NOTE

A log handler must be added to at least one logger in order to be active.

Log Handler Types

Console

A console log handler writes log messages to either the host operating system's standard out, **stdout**, or standard error, **stderr**, stream. These messages are displayed when JBoss EAP is run from a command line prompt. The messages from a console log handler are not saved unless the operating system is configured to capture the standard out or standard error stream.

File

A file log handler writes log messages to a specified file.

Periodic

A periodic log handler writes log messages to a named file until a specified period of time has elapsed. Once the time period has passed, the file is renamed by appending the specified timestamp and the handler continues to write into a newly created log file with the original name.

Size

A size log handler writes log messages to a named file until the file reaches a specified size. When the file reaches a specified size, it is renamed with a numeric suffix and the handler continues to write into a newly created log file with the original name. Each size log handler must specify the maximum number of files to be kept in this fashion.

Periodic Size

A periodic size log handler writes log messages to a named file until the file reaches the specified size or the specified time period has passed. Then, the file is renamed and the handler continues to write to a newly created log file with the original name.

This is a combination of the periodic and size log handlers and supports their combined attributes.

Syslog

A syslog handler can be used to send messages to a remote logging server. This allows multiple applications to send their log messages to the same server, where they can all be parsed together.

Socket

A socket log handler can be used to send log messages over a socket to a remote logging server. This can be a TCP or UDP socket.

Custom

A custom log handler enables you to configure new types of log handlers that have been implemented. A custom handler must be implemented as a Java class that extends **java.util.logging.Handler** and be contained in a module. You can also use a Log4J appender as a custom log handler.

Async

An async log handler is a wrapper log handler that provides asynchronous behavior for one or more other log handlers. This is useful for log handlers that may have high latency or other performance problems such as writing a log file to a network file system.

For details on configuring each of these log handlers, see the [Configuring Log Handlers](#) section.

11.3.4. Log Levels

A log level is an enumerated value that indicates the nature and severity of a log message. As a developer, you can specify the level of a given log message using the appropriate method of your chosen logging framework to send the message.

JBoss EAP supports all the log levels used by the supported application logging frameworks. The most commonly used log levels from lowest to highest are **TRACE**, **DEBUG**, **INFO**, **WARN**, **ERROR**, and **FATAL**.

Log levels are used by log categories and handlers to limit the messages they are responsible for. Each log level has an assigned numeric value that indicates its order relative to other log levels. Log categories and handlers are assigned a log level, and they only process log messages of that level or higher. For example a log handler with the level of **WARN** will only record messages of the levels **WARN**, **ERROR**, and **FATAL**.

Supported Log Levels

Log Level	Value	Description
ALL	Integer.MIN_VALUE	Provides all log messages.
FINEST	300	-
FINER	400	-
TRACE	400	TRACE level log messages provide detailed information about the running state of an application and are usually only captured during debugging.
DEBUG	500	DEBUG level log messages indicate the progress of individual requests or application activities and are usually only captured during debugging.

Log Level	Value	Description
FINE	500	-
CONFIG	700	-
INFO	800	INFO level log messages indicate the overall progress of the application. Often used for application startup, shutdown, and other major lifecycle events.
WARN	900	WARN level log messages indicate a situation that is not in error, but is not considered ideal. WARN log messages can indicate circumstances that could lead to errors in the future.
WARNING	900	-
ERROR	1000	ERROR level log messages indicate an error that has occurred that could prevent the current activity or request from completing but will not prevent the application from running.
SEVERE	1000	-
FATAL	1100	FATAL level log messages indicate events that could cause critical service failure and application shutdown and could cause JBoss EAP to shutdown.
OFF	Integer.MAX_VALUE	Does not display any log message.



NOTE

ALL is the lowest log level and includes messages of all log levels. This provides the most amount of logging.

FATAL is the highest log level and only includes messages of that level. This provides the least amount of logging.

11.3.5. Log Formatters

Formatters are used to format a log message. A formatter can be assigned to a logging handler using the **named-formatter** attribute. For more information on logging handler configuration, see [Configuring Log Handlers](#).

The logging subsystem includes four types of formatters:

- [Pattern Formatter](#)
- [JSON Formatter](#)
- [XML Formatter](#)
- [Custom Formatter](#)

Pattern Formatter

A pattern formatter is used to format log messages in plain text. In addition to using the formatter as the **named-formatter** attribute of log handlers, it can also be used as a **formatter** attribute, without the need to create the formatter resource first. See [Format Characters for Pattern Formatter](#) for more information on the pattern syntax.

See [Configure a Pattern Formatter](#) for information on how to configure a pattern formatter.

JSON Formatter

A JSON formatter is used to format log messages in JSON.

See [Configure a JSON Log Formatter](#) for information on how to configure a JSON formatter.

XML Formatter

The XML log formatter is used to format log messages in XML.

See [Configure an XML Log Formatter](#) for information on how to configure an XML log formatter.

Custom Formatter

A custom formatter to be used with handlers. Note that most log records are formatted in the printf format. Formatters may require invocation of the **org.jboss.logmanager.ExtLogRecord#getFormattedMessage()** for the message to be properly formatted.

See [Configure a Custom Log Formatter](#) for information on how to configure a custom log formatter.

11.3.6. Filter Expressions

Filter expressions, configured using the **filter-spec** attribute, are used to record log messages based on various criteria. Filter checking is always done on a raw unformatted message. You can include a filter for a logger or handler, but the logger filter takes precedence over the filter put on a handler.



NOTE

A **filter-spec** specified for the root logger is not inherited by other loggers. Instead a **filter-spec** must be specified per handler.

Table 11.3. Filter Expressions for Logging

Filter Expression	Description
accept	Accept all log messages.
deny	Deny all log messages.
not[filter expression]	Returns the inverted value of a single filter expression. For example: not(match("WFLY"))
all[filter expression]	Returns concatenated value from a comma-separated list of filter expressions. For example: all(match("WFLY"),match("WELD"))

Filter Expression	Description
<code>any[filter expression]</code>	<p>Returns one value from a comma-separated list of filter expressions. For example:</p> <p><code>any(match("WFLY"),match("WELD"))</code></p>
<code>levelChange[level]</code>	<p>Updates the log record with the specified level. For example:</p> <p><code>levelChange(WARN)</code></p>
<code>levels[levels]</code>	<p>Filters log messages with a level listed in the comma-separated list of levels. For example:</p> <p><code>levels(DEBUG,INFO,WARN,ERROR)</code></p>
<code>levelRange[minLevel,maxLevel]</code>	<p>Filters log messages within the specified level range. The <code>[</code> and <code>]</code> characters are used to indicate an inclusive level. The <code>(</code> and <code>)</code> characters are used to indicate an exclusive level. For example:</p> <ul style="list-style-type: none"> • <code>levelRange[INFO,ERROR]</code> <ul style="list-style-type: none"> ◦ The minimum level must be greater than or equal to INFO and the maximum level must be less than or equal to ERROR. • <code>levelRange[DEBUG,ERROR)</code> <ul style="list-style-type: none"> ◦ The minimum level must be greater than or equal to DEBUG and the maximum level must be less than ERROR.
<code>match["pattern"]</code>	<p>Filters log messages using the provided regular expression. For example:</p> <p><code>match("WFLY\d+")</code></p>
<code>substitute["pattern","replacement value"]</code>	<p>A filter that replaces the first match to the pattern (first argument) with the replacement text (second argument). For example:</p> <p><code>substitute("WFLY","EAP")</code></p>
<code>substituteAll["pattern","replacement value"]</code>	<p>A filter which replaces all matches of the pattern (first argument) with the replacement text (second argument). For example:</p> <p><code>substituteAll("WFLY","EAP")</code></p>

**NOTE**

When configuring the filter expression using the management CLI, be sure to escape commas and quotation marks in the filter text so that the value is correctly processed as a string. You must precede commas and quotation marks with a backslash (\) and wrap the entire expression in quotation marks. Below is an example that properly escapes **substituteAll("WFLY","YLFW")**.

```
/subsystem=logging/console-handler=CONSOLE:write-attribute(name=filter-spec,
value="substituteAll(\"WFLY\", \"YLFW\")")
```

11.3.7. Implicit Logging Dependencies

By default, the JBoss EAP **logging** subsystem adds implicit logging API dependencies to deployments. You can control whether these implicit dependencies are added to deployments by using the **add-logging-api-dependencies** attribute, which is **true** by default.

Using the management CLI, you can set the **add-logging-api-dependencies** attribute to **false** so that the implicit logging API dependencies will not be added to deployments.

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies, value=false)
```

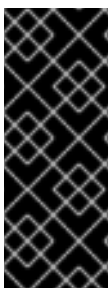
For information on the implicit dependencies for the **logging** subsystem, see the [Implicit Module Dependencies](#) section in the JBoss EAP *Development Guide*.

11.4. CONFIGURING LOG CATEGORIES

This section shows you how to configure log categories using the management CLI. You can also configure log categories using the management console by navigating to **Configuration → Subsystems → Logging → Configuration**, clicking **View**, and selecting **Categories**.

The main tasks you will perform to configure a log category are:

- [Add a new log category](#) .
- [Configure the log category settings](#).
- [Assign a log handler to the log category](#) .

**IMPORTANT**

If you are configuring this log category for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

Add a Log Category

The log category name is defined by the Java package of origin. Messages from classes in that package will be captured as long as they adhere to the other settings, for example, the log level.

```
/subsystem=logging/logger=LOG_CATEGORY:add
```

Configure Log Category Settings

Depending on your needs, you may need to set one or more of the following log category attributes. For a full list of available log category attributes and their descriptions, see [Log Category Attributes](#).

- Set the log level.
Set the appropriate log level for the log category. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=level,value=LEVEL)
```

- Set whether this category should use the log handlers of the root logger.
By default, log categories will use the handlers of the root logger in addition to its own. Set the **use-parent-handlers** attribute to **false** if the log category should use only its assigned handlers.

```
/subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=use-parent-handlers,value=USE_PARENT_HANDLERS)
```

- Set the filter expression.
Set the expression for filtering log messages for the log category. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=filter-spec,value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign a Handler

Assign a log handler to the log category.

```
/subsystem=logging/logger=LOG_CATEGORY:add-handler(name=LOG_HANDLER_NAME)
```

Remove a Log Category

A log category can be removed with the **remove** operation.

```
/subsystem=logging/logger=LOG_CATEGORY:remove
```

11.5. CONFIGURING LOG HANDLERS

Log handlers define how captured log messages are recorded. See the appropriate section for configuring the type of log handler that you need.

- [Console log handler](#)
- [File log handler](#)
- [Periodic rotating log handler](#)
- [Size rotating log handler](#)
- [Periodic size rotating log handler](#)

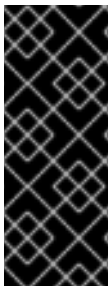
- [Syslog handler](#)
- [Socket log handler](#)
- [Custom log handler](#)
- [Async log handler](#)

11.5.1. Configure a Console Log Handler

This section shows you how to configure a console log handler using the management CLI. You can also configure console log handlers using the management console by navigating to **Configuration** → **Subsystems** → **Logging** → **Configuration**, clicking **View**, and selecting **Handler** → **Console Handler**.

The main tasks you will perform to configure a console log handler are:

- [Add a new console log handler](#) .
- [Configure the console log handler settings](#).
- [Assign the console log handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be `/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/` instead of `/subsystem=logging/`.

Additionally, if you are running in a managed domain, precede the commands with `/profile=PROFILE_NAME.`

Add a Console Log Handler

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:add
```

Configure Console Log Handler Settings

Depending on your needs, you may need to set one or more of the following console log handler attributes. For a full list of available console log handler attributes and their descriptions, see [Console Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the target.
Set the target for the handler, which can be one of **System.out**, **System.err**, or **console**. The default is **System.out**.

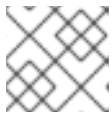
```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=target,value=TARGET)
```

- Set the encoding.
Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a [saved formatter](#).

- Set auto flush.
Set whether to automatically flush after each write. The default value is **true**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Console Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the console log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=CONSOLE_HANDLER_NAME)
```

The following management CLI command assigns the console log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=CONSOLE_HANDLER_NAME)
```

Remove a Console Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:remove
```

11.5.2. Configure a File Log Handler

This section shows you how to configure a file log handler using the management CLI. You can also configure file log handlers using the management console by navigating to **Configuration** → **Subsystems** → **Logging** → **Configuration**, clicking **View**, and selecting **Handler** → **File Handler**.

The main tasks you will perform to configure a file log handler are:

- [Add a new file log handler](#) .
- [Configure the file log handler settings](#).
- [Assign the file log handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME.**

Add a File Log Handler

When adding a file log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example **my-log.log**. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example **jboss.server.log.dir**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH})
```

Configure File Log Handler Settings

Depending on your needs, you may need to set one or more of the following file log handler attributes. For a full list of available file log handler attributes and their descriptions, see [File Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the append behavior.
By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **false** to have the file overwritten upon server restart.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- Set the encoding.
Set the encoding for the handler, for example, **utf-8**.


```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a [saved formatter](#).

- Set auto flush.
Set whether to automatically flush after each write. The default value is **true**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match(\"WFLY\"))**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=filter-spec,value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the File Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the file log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=FILE_HANDLER_NAME)
```

The following management CLI command assigns the file log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=FILE_HANDLER_NAME)
```

Remove a File Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:remove
```

11.5.3. Configure a Periodic Rotating Log Handler

This section shows you how to configure a periodic rotating log handler using the management CLI. You can also configure periodic log handlers using the management console by navigating to **Configuration** → **Subsystems** → **Logging** → **Configuration**, clicking **View**, and selecting **Handler** → **Periodic Handler**.

The main tasks you will perform to configure a periodic log handler are:

- [Add a new periodic log handler](#) .
- [Configure the periodic log handler settings](#) .
- [Assign the periodic log handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be `/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/` instead of `/subsystem=logging/`.

Additionally, if you are running in a managed domain, precede the commands with `/profile=PROFILE_NAME.`

Add a Periodic Log Handler

When adding a periodic log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example `my-log.log`. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example `jboss.server.log.dir`.

You must also set the suffix for rotated logs using the **suffix** attribute. This must be in a format that can be understood by `java.text.SimpleDateFormat`, for example `.yyyy-MM-dd-HH`. The period of the rotation is automatically calculated based on this suffix.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH},suffix=SUFFIX)
```

Configure Periodic Log Handler Settings

Depending on your needs, you may need to set one or more of the following periodic log handler attributes. For a full list of available periodic log handler attributes and their descriptions, see [Periodic Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the append behavior.
By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **false** to have the file overwritten upon server restart.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- Set the encoding.

Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.

Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a [saved formatter](#).

- Set auto flush.

Set whether to automatically flush after each write. The default value is **true**.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.

Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the ***FILTER_EXPRESSION*** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Periodic Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the periodic log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=PERIODIC_HANDLER_NAME)
```

The following management CLI command assigns the periodic log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=PERIODIC_HANDLER_NAME)
```

Remove a Periodic Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

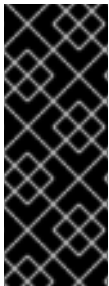
```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:remove
```

11.5.4. Configure a Size Rotating Log Handler

This section shows you how to configure a size rotating log handler using the management CLI. You can also configure size log handlers using the management console by navigating to **Configuration** → **Subsystems** → **Logging** → **Configuration**, clicking **View**, and selecting **Handler** → **Size Handler**.

The main tasks you will perform to configure a size log handler are:

- [Add a new size log handler](#) .
- [Configure the size log handler settings](#) .
- [Assign the size log handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME.**

Add a Size Log Handler

When adding a size log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example **my-log.log**. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example **jboss.server.log.dir**.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH})
```

Configure Size Log Handler Settings

Depending on your needs, you may need to set one or more of the following size log handler attributes. For a full list of available size log handler attributes and their descriptions, see [Size Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the suffix for rotated logs.
Set the suffix string, which is in a format which can be understood by **java.text.SimpleDateFormat**, for example **.yyyy-MM-dd-HH**. The period of the rotation is automatically calculated based on this suffix.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=suffix,value=SUFFIX)
```

- Set the rotation size.

Set the maximum size that the file can reach before being rotated. The default is **2m** for 2 megabytes.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=rotate-size, value=ROTATE_SIZE)
```

- Set the maximum number of backup logs to keep.
Set the number of backups to keep. The default is **1**.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=max-backup-index, value=MAX_BACKUPS)
```

- Set whether to rotate the log on boot.
By default, a new log file is not created on server restart. You can set this to **true** to rotate the log on server restart.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=rotate-on-boot, value=ROTATE_ON_BOOT)
```

- Set the append behavior.
By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **false** to have the file overwritten upon server restart.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- Set the encoding.
Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a [saved formatter](#).

- Set auto flush.
Set whether to automatically flush after each write. The default value is **true**.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.

Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the ***FILTER_EXPRESSION*** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Size Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the size log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SIZE_HANDLER_NAME)
```

The following management CLI command assigns the size log handler to a logger whose name is specified by ***CATEGORY***.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SIZE_HANDLER_NAME)
```

Remove a Size Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

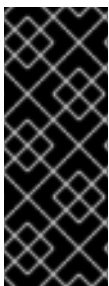
```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:remove
```

11.5.5. Configure a Periodic Size Rotating Log Handler

This section shows you how to configure a periodic size rotating log handler using the management CLI. You can also configure periodic size log handlers using the management console by navigating to **Configuration → Subsystems → Logging → Configuration**, clicking **View**, and selecting **Handler → Periodic Size Handler**.

The main tasks you will perform to configure a periodic size log handler are:

- [Add a new periodic size log handler](#) .
- [Configure the periodic size log handler settings](#) .
- [Assign the periodic size log handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add a Periodic Size Log Handler

When adding a periodic size log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example **my-log.log**. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example **jboss.server.log.dir**.

You must also set the suffix for rotated logs using the **suffix** attribute. This must be in a format that can be understood by **java.text.SimpleDateFormat**, for example **.yyyy-MM-dd-HH**. The period of the rotation is automatically calculated based on this suffix.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH},suffix=SUFFIX)
```

Configure Periodic Size Log Handler Settings

Depending on your needs, you may need to set one or more of the following periodic size log handler attributes. For a full list of available periodic size log handler attributes and their descriptions, see [Periodic Size Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the rotation size.
Set the maximum size that the file can reach before being rotated. The default is **2m** for 2 megabytes.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=rotate-size,value=ROTATE_SIZE)
```

- Set the maximum number of backup logs to keep.
Set the number of backups to keep. The default is **1**.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=max-backup-index,value=MAX_BACKUPS)
```

- Set whether to rotate the log on boot.
By default, a new log file is not created on server restart. You can set this to **true** to rotate the log on server restart.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=rotate-on-boot,value=ROTATE_ON_BOOT)
```

- Set the append behavior.
By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **false** to have the file overwritten upon server restart.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- Set the encoding.
Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:write-
attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:write-
attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a [saved formatter](#).

- Set auto flush.
Set whether to automatically flush after each write. The default value is **true**.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:write-
attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=filter-spec,
value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Periodic Size Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the periodic size log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=PERIODIC_SIZE_HANDLER_NAME)
```

The following management CLI command assigns the periodic size log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=PERIODIC_SIZE_HANDLER_NAME)
```

Remove a Periodic Size Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

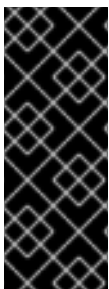
```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:remove
```

11.5.6. Configure a Syslog Handler

This section shows you how to configure a syslog handler using the management CLI, which can be used to send messages to a remote logging server that supports the Syslog protocol, either RFC-3164 or RFC-5424. You can also configure syslog handlers using the management console by navigating to **Configuration → Subsystems → Logging → Configuration**, clicking **View**, and selecting **Handler → Syslog Handler**.

The main tasks you will perform to configure a syslog handler are:

- [Add a new syslog handler](#) .
- [Configure the syslog handler settings](#) .
- [Assign the syslog handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=*LOGGING_PROFILE_NAME*** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=*PROFILE_NAME***.

Add a Syslog Handler

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:add
```

Configure Syslog Handler Settings

Depending on your needs, you may need to set one or more of the following syslog handler attributes. For a full list of available syslog handler attributes and their descriptions, see [Syslog Handler Attributes](#) .

- Set the log level for the handler. The default level is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the name of the application that is logging. The default name is **java**.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=app-name,value=APP_NAME)
```

- Set the address of the syslog server. The default address is **localhost**.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=server-address,value=SERVER_ADDRESS)
```

- Set the port of the syslog server. The default port is **514**.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=port,value=PORT)
```

- Set the syslog format, as defined by an RFC specification. The default format is **RFC5424**.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=syslog-format,value=SYSLOG_FORMAT)
```

- Specify the **named-formatter** attribute to format the message of the syslog payload.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=named-formatter,value=FORMATTER_NAME)
```

Assign the Syslog Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the syslog handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SYSLOG_HANDLER_NAME)
```

The following management CLI command assigns the syslog handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SYSLOG_HANDLER_NAME)
```

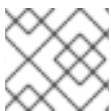
Remove a Syslog Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:remove
```

11.5.7. Configure a Socket Log Handler

This section shows you how to configure a socket log handler using the management CLI, which can be used to send messages over a socket. This can be a TCP or UDP socket. You can also configure socket log handlers using the management console by navigating to **Configuration → Subsystems → Logging → Configuration**, clicking **View**, and selecting **Handler → Socket Handler**.

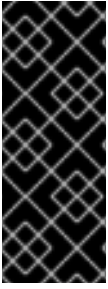


NOTE

If the server is started in admin-only mode, log messages are discarded.

The main tasks you will perform to configure a socket log handler are:

- [Add a socket binding](#).
- [Add a log formatter](#).
- [Add the socket log handler](#) and [configure its settings](#).
- [Assign the socket log handler to a logger](#).



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be `/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/` instead of `/subsystem=logging/`.

Additionally, if you are running in a managed domain, precede the commands with `/profile=PROFILE_NAME.`

Add the Socket Binding

Define either a **remote-destination-outbound-socket-binding** or **local-destination-outbound-socket-binding** as the [socket binding](#) to use.

```
/socket-binding-group=SOCKET_BINDING_GROUP/remote-destination-outbound-socket-binding=SOCKET_BINDING_NAME:add(host=HOST, port=PORT)
```

Add the Log Formatter

Define a [log formatter](#) to use, such as a JSON formatter.

```
/subsystem=logging/json-formatter=FORMATTER:add
```

Add the Socket Log Handler

When adding the socket log handler, you must specify the socket binding and formatter to use.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:add(outbound-socket-binding-ref=SOCKET_BINDING_NAME,named-formatter=FORMATTER)
```

Configure Socket Log Handler Settings

Depending on your needs, you may need to set one or more of the following socket log handler attributes. For a full list of available socket log handler attributes and their descriptions, see [Socket Log Handler Attributes](#).

- Set the protocol.
Set the protocol to use. The default is **TCP**.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=protocol,value=PROTOCOL)
```

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```



NOTE

During server startup, log messages handled by a socket log handler are queued until the socket binding is configured and the **logging** subsystem is initialized. If the log level is set to a low level, such as **TRACE** or **DEBUG**, this can result in large memory consumption during startup.

- Set the encoding.
Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set auto flush.
Set whether to automatically flush after each write. The default value is **true**.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the ***FILTER_EXPRESSION*** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=filter-spec,value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Socket Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the socket log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SOCKET_HANDLER_NAME)
```

The following management CLI command assigns the socket log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SOCKET_HANDLER_NAME)
```

Remove a Socket Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:remove
```

Send Log Messages Over a Socket Using SSL/TLS

The following steps show an example of how to set up a socket log handler to send log messages over a socket using the **SSL_TCP** protocol. This example configures a keystore, trust manager, and client SSL context in the **elytron** subsystem to be used by the socket log handler. Log messages from the root logger are sent over the specified socket, formatted by a JSON formatter.

For more information on configuring Elytron components, see [Elytron Subsystem](#) in *How to Configure Server Security* for JBoss EAP.

1. Configure the Elytron settings.
 - a. Add the keystore.

■

```
/subsystem=elytron/key-store=log-server-ks:add(path=/path/to/keystore.jks, type=JKS,
credential-reference={clear-text=mypassword})
```

- b. Add the trust manager.

```
/subsystem=elytron/trust-manager=log-server-tm:add(key-store=log-server-ks)
```

- c. Add the client SSL context.

```
/subsystem=elytron/client-ssl-context=log-server-context:add(trust-manager=log-server-
tm, protocols=["TLSv1.2"])
```

2. Add the socket binding.

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=log-
server:add(host=localhost, port=4560)
```

3. Add a JSON formatter.

```
/subsystem=logging/json-formatter=json:add
```

4. Add the socket log handler.

```
/subsystem=logging/socket-handler=log-server-handler:add(named-formatter=json,
level=INFO, outbound-socket-binding-ref=log-server, protocol=SSL_TCP, ssl-context=log-
server-context)
```

5. Assign the log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=log-server-handler)
```

11.5.8. Configure a Custom Log Handler

This section shows you how to configure a custom log handler using the management CLI. You can also configure custom log handlers using the management console by navigating to **Configuration** → **Subsystems** → **Logging** → **Configuration**, clicking **View**, and selecting **Handler** → **Custom Handler**.

The main tasks you will perform to configure a custom log handler are:

- [Add a new custom log handler](#) .
- [Configure the custom log handler settings](#).
- [Assign the custom log handler to a logger](#) .



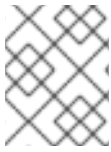
IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

Add a Custom Log Handler

When adding a custom log handler, you must specify the Java class of the handler and the JBoss EAP module in which it is contained. The class must extend `java.util.logging.Handler`.



NOTE

You must have already [created a module](#) containing the custom logger or this command will fail.

```
/subsystem=logging/custom-  
handler=CUSTOM_HANDLER_NAME:add(class=CLASS_NAME,module=MODULE_NAME)
```

Configure Custom Log Handler Settings

Depending on your needs, you may need to set one or more of the following custom log handler attributes. For a full list of available custom log handler attributes and their descriptions, see [Custom Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-  
attribute(name=level,value=LEVEL)
```

- Set the properties.
Set the necessary properties for the log handler. The properties must be accessible using a setter method.

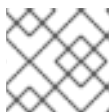
```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-  
attribute(name=properties.PROPERTY_NAME,value=PROPERTY_VALUE)
```

- Set the encoding.
Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-  
attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-  
attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a [saved formatter](#).

- Set the filter expression.
Set the expression for filtering log messages for the handler. Be sure to escape any commas

and quotation marks and surround with quotation marks. For example, the ***FILTER_EXPRESSION*** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Custom Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the custom log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=CUSTOM_HANDLER_NAME)
```

The following management CLI command assigns the custom log handler to a logger whose name is specified by ***CATEGORY***.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=CUSTOM_HANDLER_NAME)
```

Remove a Custom Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

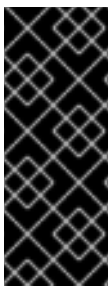
```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:remove
```

11.5.9. Configure an Async Log Handler

This section shows you how to configure an async log handler using the management CLI. You can also configure async log handlers using the management console by navigating to **Configuration → Subsystems → Logging → Configuration**, clicking **View**, and selecting **Handler → Async Handler**.

The main tasks you will perform to configure an async log handler are:

- [Add a new async log handler](#) .
- [Add sub-handlers to the async log handler](#) .
- [Configure the async log handler settings](#) .
- [Assign the async log handler to a logger](#) .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add an Async Log Handler

When adding an async log handler, you must specify the queue length. This is the maximum number of log requests that can be held in queue.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:add(queue-length=QUEUE_LENGTH)
```

Add a Sub-handler

You can add one or more handlers as sub-handlers for this async log handler. Note that the handlers must already exist in the configuration or this command will fail.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:add-handler(name=HANDLER_NAME)
```

Configure Async Log Handler Settings

Depending on your needs, you may need to set one or more of the following async log handler attributes. For a full list of available async log handler attributes and their descriptions, see [Async Log Handler Attributes](#).

- Set the log level.
Set the appropriate log level for the handler. The default is **ALL**. See [Log Levels](#) for all available options.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the overflow action.
Set the action to take when overflowing. The default value is **BLOCK**, which means that threads will block in the event of a full queue. You can change this value to **DISCARD**, which means that log messages will be discarded in the event of a full queue.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-attribute(name=overflow-action,value=OVERFLOW_ACTION)
```

- Set the filter expression.
Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the ***FILTER_EXPRESSION*** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

See the [Filter Expressions](#) section for more information on available filter expressions.

Assign the Async Log Handler to a Logger

In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the async log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=ASYNC_HANDLER_NAME)
```

The following management CLI command assigns the async log handler to a logger whose name is specified by **CATEGORY**.

–


```
/subsystem=logging/logger=CATEGORY:add-handler(name=ASYNC_HANDLER_NAME)
```

Remove an Async Log Handler

A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger.

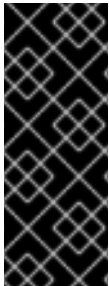
```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:remove
```

11.6. CONFIGURING THE ROOT LOGGER

The root logger captures all log messages, of the specified log level or higher, sent to the server that are not captured by a log category.

This section shows you how to configure the root logger using the management CLI. You can also configure the root logger using the management console by navigating to **Configuration** → **Subsystems** → **Logging** → **Configuration**, clicking **View**, and selecting **Root Logger**.

Configure the Root Logger



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be `/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME` instead of `/subsystem=logging/`.

Additionally, if you are running in a managed domain, precede the commands with `/profile=PROFILE_NAME`.

1. Assign log handlers to the root logger.
Add a log handler.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=LOG_HANDLER_NAME)
```

Remove a log handler.

```
/subsystem=logging/root-logger=ROOT:remove-handler(name=LOG_HANDLER_NAME)
```

2. Set the log level.

```
/subsystem=logging/root-logger=ROOT:write-attribute(name=level,value=LEVEL)
```

For a full list of available root logger attributes and their descriptions, see [Root Logger Attributes](#).

11.7. CONFIGURING LOG FORMATTERS

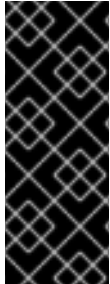
A log formatter defines the appearance of log messages from that handler. The **logging** subsystem allows you to configure the following types of log formatters:

- [Pattern Formatter](#)
- [JSON Log Formatter](#)

- [XML Log Formatter](#)
- [Custom Log Formatter](#)

11.7.1. Configure a Pattern Formatter

You can create a named pattern formatter that can be used across log handlers to format log messages.



IMPORTANT

If you are configuring this log formatter for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME.**

Create a Pattern Formatter

When defining a pattern formatter, you provide a pattern string to use to format log messages. For more information on the pattern syntax, see [Format Characters for Pattern Formatter](#).

```
/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:add(pattern=PATTERN)
```

For example, the default configuration uses the following log formatter string for logging messages to the server log: **%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. This creates log messages formatted like the one below.

```
2016-03-18 15:49:32,075 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin
console listening on http://127.0.0.1:9990
```

You can also define a color map to assign a color to different log levels. The format is a comma-separated list of **LEVEL:COLOR**.

- Valid levels: **finest, finer, fine, config, trace, debug, info, warning, warn, error, fatal, severe**
- Valid colors: **black, green, red, yellow, blue, magenta, cyan, white, brightblack, brightred, brightgreen, brightblue, brightyellow, brightmagenta, brightcyan, brightwhite**

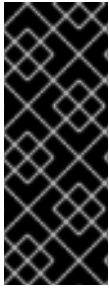
```
/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:write-attribute(name=color-
map,value="LEVEL:COLOR,LEVEL:COLOR")
```

You can also configure pattern log formatters using the management console:

1. Open the management console in a browser.
2. Choose **Configuration → Subsystems → Logging**.
3. Select **Configuration** and then click **View**.
4. Select **Formatter** and then choose the **Pattern Formatter** option.

11.7.2. Configure a JSON Log Formatter

You can create a JSON log formatter to format log messages in JSON.



IMPORTANT

If you are configuring this log formatter for a logging profile, the start of the command would be `/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/` instead of `/subsystem=logging/`.

Additionally, if you are running in a managed domain, precede the commands with `/profile=PROFILE_NAME.`

Add a JSON Log Formatter

```
/subsystem=logging/json-formatter=JSON_FORMATTER_NAME:add(pretty-print=true, exception-output-type=formatted)
```

This creates log messages formatted like the one below.

```
{
  "timestamp": "2018-10-18T13:53:43.031-04:00",
  "sequence": 62,
  "loggerClassName": "org.jboss.as.server.logging.ServerLogger_$logger",
  "loggerName": "org.jboss.as",
  "level": "INFO",
  "message": "WFLYSRV0025: JBoss EAP 7.3.0.GA (WildFly Core 10.0.0.Final-redhat-20190924)
started in 5672ms - Started 495 of 679 services (331 services are lazy, passive or on-demand)",
  "threadName": "Controller Boot Thread",
  "threadId": 22,
  "mdc": {
  },
  "ndc": "",
  "hostName": "localhost.localdomain",
  "processName": "jboss-modules.jar",
  "processId": 7461
}
```

Add a Logstash JSON Log Formatter

```
/subsystem=logging/json-formatter=logstash:add(exception-output-type=formatted, key-overrides=[timestamp="@timestamp"], meta-data=[@version=1])
```

You can use the JSON formatter attributes as explained below:

- The **key-overrides** attribute can be used to override the names of the keys defined.
- The exceptions can be formatted as objects by setting the **exception-output-type** attribute to **formatted**.
- The exception stack trace can be included by setting the **exception-output-type** attribute to **detailed**.
- The exceptions can be formatted as objects and stack traces can be included by setting the **exception-output-type** to **detailed-and-formatted**.
- The metadata can be added to log records using the **meta-data** attribute.

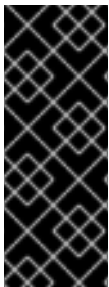
For more information on the JSON formatter attributes, see [JSON Log Formatter Attributes](#).

You can also configure JSON log formatters using the management console:

1. Open the management console in a browser.
2. Choose **Configuration** → **Subsystems** → **Logging**.
3. Select **Configuration** and then click **View**.
4. Select **Formatter** and then choose the **JSON Formatter** option.

11.7.3. Configure an XML Log Formatter

You can create an XML log formatter to format log messages in XML.



IMPORTANT

If you are configuring this log formatter for a logging profile, the start of the command would be `/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/` instead of `/subsystem=logging/`.

Additionally, if you are running in a managed domain, precede the commands with `/profile=PROFILE_NAME.`

Add an XML Log Formatter

```
/subsystem=logging/xml-formatter=XML_FORMATTER_NAME:add(pretty-print=true, exception-
output-type=detailed-and-formatted)
```

This creates log message formatted like the one below.

```
<record>
  <timestamp>2018-10-18T13:55:53.419-04:00</timestamp>
  <sequence>62</sequence>
  <loggerClassName>org.jboss.as.server.logging.ServerLogger_$logger</loggerClassName>
  <loggerName>org.jboss.as</loggerName>
  <level>INFO</level>
  <message>WFLYSRV0025: JBoss EAP 7.3.0.GA (WildFly Core 10.0.0.Final-redhat-20190924)
started in 6271ms - Started 495 of 679 services (331 services are lazy, passive or on-
demand)</message>
  <threadName>Controller Boot Thread</threadName>
  <threadId>22</threadId>
  <mdc>
  </mdc>
  <ndc></ndc>
  <hostName>localhost.localdomain</hostName>
  <processName>jboss-modules.jar</processName>
  <processId>7790</processId>
</record>
```

Add a Key Override XML Log Formatter

```
/subsystem=logging/xml-formatter=XML_FORMATTER_NAME:add(pretty-print=true, print-
namespace=true, namespace-uri="urn:custom:1.0", key-overrides={message=msg,
record=logRecord, timestamp=date}, print-details=true)
```

You can use the XML formatter attributes as explained below:

- The **key-overrides** attribute can be used to override the names of the keys defined.
- The exceptions can be formatted as objects by setting the **exception-output-type** attribute to **formatted**.
- The exception stack trace can be included by setting the **exception-output-type** attribute to **detailed**.
- The exceptions can be formatted as objects and stack traces can be included by setting the **exception-output-type** to **detailed-and-formatted**.
- The metadata can be added to log records using the **meta-data** attribute.

For more information on the XML formatter attributes, see [XML Log Formatter Attributes](#).

You can also configure XML log formatters using the management console:

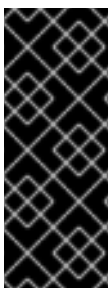
1. Open the management console in a browser.
2. Choose **Configuration** → **Subsystems** → **Logging**.
3. Select **Configuration** and then click **View**.
4. Select **Formatter** and then choose the **XML Formatter** option.

11.7.4. Configure a Custom Log Formatter

You can create a custom log formatter that can be used across log handlers to format log messages.

This section shows you how to configure a custom log formatter using the management CLI.

Configure a Custom Log Formatter



IMPORTANT

If you are configuring this log formatter for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

1. Add the custom log formatter.
When adding a custom log formatter, you must specify the Java class of the formatter and the JBoss EAP module in which it is contained. The class must extend **java.util.logging.Formatter**.



NOTE

You must have already created a module containing the custom formatter or this command will fail.

```
/subsystem=logging/custom-  
formatter=CUSTOM_FORMATTER_NAME:add(class=CLASS_NAME,  
module=MODULE_NAME)
```

2. Set the necessary properties for the log formatter.
The properties must be accessible using a setter method.

```
/subsystem=logging/custom-formatter=CUSTOM_FORMATTER_NAME:write-  
attribute(name=properties.PROPERTY_NAME,value=PROPERTY_VALUE)
```

3. Assign the custom formatter to a log handler.
The following management CLI command assigns a custom formatter to be used by a periodic rotating file handler.

```
/subsystem=logging/periodic-rotating-file-handler=FILE_HANDLER_NAME:write-  
attribute(name=named-formatter, value=CUSTOM_FORMATTER_NAME)
```

Example Custom XML Formatter

The following example configures a custom XML formatter. It uses the `java.util.logging.XMLFormatter` class provided in the `org.jboss.logmanager` module and assigns it to the console log handler.

```
/subsystem=logging/custom-formatter=custom-xml-  
formatter:add(class=java.util.logging.XMLFormatter, module=org.jboss.logmanager)  
/subsystem=logging/console-handler=CONSOLE:write-attribute(name=named-formatter,  
value=custom-xml-formatter)
```

A log message using this formatter would be formatted as below.

```
<record>  
  <date>2016-03-23T12:58:13</date>  
  <millis>1458752293091</millis>  
  <sequence>93963</sequence>  
  <logger>org.jboss.as</logger>  
  <level>INFO</level>  
  <class>org.jboss.as.server.BootstrapListener</class>  
  <method>logAdminConsole</method>  
  <thread>22</thread>  
  <message>WFLYSRV0051: Admin console listening on http://%s:%d</message>  
  <param>127.0.0.1</param>  
  <param>9990</param>  
</record>
```

Configure a Custom Log Formatter Using the Management Console

You can also configure log formatters using the management console.

1. Open the management console in a browser.
2. Choose **Configuration** → **Subsystems** → **Logging**.
3. Select **Configuration** and then click **View**.
4. Select **Formatter** and then choose the **Custom Formatter** option.

11.8. ABOUT APPLICATION LOGGING

Logging for applications can be configured using the JBoss EAP **logging** subsystem or on a per-deployment basis.

See [About the Logging Subsystem](#) for using JBoss EAP log categories and handlers for capturing log messages.

For more information on application logging, such as supported application logging frameworks and configuring per-deployment logging, see the [Logging](#) chapter in the JBoss EAP *Development Guide*.

11.8.1. Per-deployment Logging

Per-deployment logging allows a developer to configure the logging configuration for their application in advance. When the application is deployed, logging begins according to the defined configuration. The log files created through this configuration contain information only about the behavior of the application.



NOTE

If the per-deployment logging configuration is not done, the configuration from **logging** subsystem is used for all the applications as well as the server.

This approach has advantages and disadvantages over using system-wide logging. An advantage is that the administrator of the JBoss EAP instance does not need to configure any other logging than the server logging. A disadvantage is that the per-deployment logging configuration is read only on server startup, and so cannot be changed at runtime.

For instructions on using per-deployment logging in your applications, see [Add Per-deployment Logging to an Application](#) in the JBoss EAP *Development Guide*.

11.8.1.1. Disable Per-deployment Logging

You can disable per-deployment logging in one of the following ways:

- Set the **use-deployment-logging-config** attribute to **false**.
The **use-deployment-logging-config** attribute controls whether or not your deployment is scanned for per-deployment logging. This defaults to **true** by default. You can set this attribute to **false** to disable per-deployment logging.

```
/subsystem=logging:write-attribute(name=use-deployment-logging-config,value=false)
```

- Exclude the **logging** subsystem using a **jboss-deployment-structure.xml** file.
For instructions, see [Exclude a Subsystem from a Deployment](#) in the JBoss EAP *Development Guide*.

11.8.2. Logging Profiles

Logging profiles are independent sets of logging configurations that can be assigned to deployed applications. As with the regular **logging** subsystem, a logging profile can define handlers, categories, and a root logger, but it cannot refer to configurations in other profiles or the main **logging** subsystem. The design of logging profiles mimics the **logging** subsystem for ease of configuration.

Logging profiles allow administrators to create logging configurations that are specific to one or more

applications without affecting any other logging configurations. Because each profile is defined in the server configuration, the logging configuration can be changed without requiring that the affected applications be redeployed.

Each logging profile can have:

- A unique name. This value is required.
- Any number of log handlers.
- Any number of log categories.
- Up to one root logger.

An application can specify a logging profile to use in its **MANIFEST.MF** file, using the **Logging-Profile** attribute.

11.8.2.1. Configure a Logging Profile

A logging profile can be configured with log handlers, categories, and a root logger. Configuring a logging profile uses the same syntax as configuring the **logging** subsystem, except for the following differences:

- The root configuration path is **/subsystem=logging/logging-profile=NAME**.
- A logging profile cannot contain other logging profiles.
- The **logging** subsystem has the following attributes that are not available for a logging profile:
 - **add-logging-api-dependencies**
 - **use-deployment-logging-config**

Creating and Configuring a Logging Profile

The following procedure uses the management CLI to create a logging profile and set a file handler and logger category. Logging profiles can also be configured using the management console by navigating to **Configuration → Subsystems → Logging → Logging Profiles**.

1. Create the logging profile.

```
/subsystem=logging/logging-profile=PROFILE_NAME:add
```

2. Create the file handler.

```
/subsystem=logging/logging-profile=PROFILE_NAME/file-  
handler=FILE_HANDLER_NAME:add(file={path=>"LOG_NAME.log", "relative-  
to"=>"jboss.server.log.dir"})
```

```
/subsystem=logging/logging-profile=PROFILE_NAME/file-  
handler=FILE_HANDLER_NAME:write-attribute(name="level", value="DEBUG")
```

See [File Log Handler Attributes](#) for the list of file handler attributes.

3. Create the logger category.


```
/subsystem=logging/logging-
profile=PROFILE_NAME/logger=CATEGORY_NAME:add(level=TRACE)
```

See [Log Category Attributes](#) for the list of log category attributes.

4. Assign the file handler to the category.

```
/subsystem=logging/logging-profile=PROFILE_NAME/logger=CATEGORY_NAME:add-
handler(name="FILE_HANDLER_NAME")
```

You can then set the logging profile to use by an application in its **MANIFEST.MF** file. For more information, see [Specify a Logging Profile in an Application](#) in the JBoss EAP *Development Guide*.

11.8.2.2. Example Logging Profile Configuration

This example shows the configuration of a logging profile and the application that uses it. It shows the management CLI commands, the resulting XML, and the application's **MANIFEST.MF** file.

The example logging profile has the following characteristics:

- The name is **accounts-app-profile**.
- The log category is **com.company.accounts.ejbs**.
- The log level **TRACE**.
- The log handler is a file handler using the file **ejb-trace.log**.

Management CLI Session

```
/subsystem=logging/logging-profile=accounts-app-profile:add

/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:add(file=
{path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})

/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:write-
attribute(name="level", value="DEBUG")

/subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add(level=TRACE)

/subsystem=logging/logging-profile=accounts-app-profile/logger=com.company.accounts.ejbs:add-
handler(name="ejb-trace-file")
```

XML Configuration

```
<logging-profiles>
  <logging-profile name="accounts-app-profile">
    <file-handler name="ejb-trace-file">
      <level name="DEBUG"/>
      <file relative-to="jboss.server.log.dir" path="ejb-trace.log"/>
    </file-handler>
    <logger category="com.company.accounts.ejbs">
      <level name="TRACE"/>
    </logger>
  </logging-profile>
</logging-profiles>
```

```

    <handlers>
      <handler name="ejb-trace-file"/>
    </handlers>
  </logger>
</logging-profile>
</logging-profiles>

```

Application MANIFEST.MF file

```

Manifest-Version: 1.0
Logging-Profile: accounts-app-profile

```

11.8.3. Viewing Deployment Logging Configuration

You can obtain information about the logging configuration for a particular deployment using the following management CLI command.

```
/deployment=DEPLOYMENT_NAME/subsystem=logging/configuration=CONFIG:read-resource
```

The logging configuration value, **CONFIG**, for a deployment can be one of three values:

- **default**, if the deployment is using the [logging subsystem](#). This will output the **logging** subsystem configuration.
- **profile-PROFILE_NAME**, if the deployment is using a [logging profile](#) defined in the **logging** subsystem. This will output the logging profile configuration.
- The path to the configuration file being used, for example, **myear.ear/META-INF/logging.properties**.

For example, the below management CLI command displays the configuration for the **MYPROFILE** logging profile, which is used by the specified deployment.

```
/deployment=mydeployment.war/subsystem=logging/configuration=profile-MYPROFILE:read-resource(recursive=true,include-runtime=true)
```

This will output the following information.

```

{
  "outcome" => "success",
  "result" => {
    "error-manager" => undefined,
    "filter" => undefined,
    "formatter" => {
      "MYFORMATTER" => {
        "class-name" => "org.jboss.logmanager.formatters.PatternFormatter",
        "module" => undefined,
        "properties" => {"pattern" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n"}
      }
    },
    "handler" => {
      "MYPERIODIC" => {
        "class-name" => "org.jboss.logmanager.handlers.PeriodicRotatingFileHandler",
        "encoding" => undefined,

```

```

        "error-manager" => undefined,
        "filter" => undefined,
        "formatter" => "MYFORMATTER",
        "handlers" => [],
        "level" => "ALL",
        "module" => undefined,
        "properties" => {
            "append" => "true",
            "autoFlush" => "true",
            "enabled" => "true",
            "suffix" => ".yyyy-MM-dd",
            "fileName" => "EAP_HOME/standalone/log/deployment.log"
        }
    },
    "logger" => {"MYCATEGORY" => {
        "filter" => undefined,
        "handlers" => [],
        "level" => "DEBUG",
        "use-parent-handlers" => true
    }},
    "pojo" => undefined
}
}

```

You could also use a recursive **read-resource** operation to read the logging configuration and other information about a deployment.

```

/deployment=DEPLOYMENT_NAME/subsystem=logging:read-resource(include-runtime=true,
recursive=true)

```

11.9. TUNING THE LOGGING SUBSYSTEM

For tips on monitoring and optimizing performance for the **logging** subsystem, see the [Logging Subsystem Tuning](#) section of the *Performance Tuning Guide*.

CHAPTER 12. DATASOURCE MANAGEMENT

12.1. ABOUT JBOSS EAP DATASOURCES

About JDBC

The JDBC API is the standard that defines how databases are accessed by Java applications. An application configures a datasource that references a JDBC driver. Application code can then be written against the driver, rather than the database. The driver converts the code to the database language. This means that if the correct driver is installed, an application can be used with any supported database.

For more information, see the [JDBC 4.0 specification](#).

Supported Databases

See [JBoss EAP supported configurations](#) for the list of JDBC-compliant databases supported by JBoss EAP 7.

Datasource Types

The two general types of resources are referred to as *datasources* and *XA datasources*.

Non-XA datasources

Used for applications that do not use transactions, or applications that use transactions with a single database.

XA datasources

Used by applications that use multiple databases or other XA resources as part of one XA transaction. XA datasources introduce additional overhead.

You specify which type of datasource to use when creating the datasource using the JBoss EAP management interfaces.

The ExampleDS datasource

JBoss EAP ships with an example datasource configuration, *ExampleDS*, which is provided to demonstrate how datasources are defined. This datasource uses an H2 database, which is a lightweight, relational database management system that provides developers with the ability to quickly build applications.



WARNING

The *ExampleDS* datasource and the H2 database should *not* be used in a production environment. This is a very small, self-contained datasource that supports all of the standards needed for testing and building applications, but is not robust or scalable enough for production use.

12.2. JDBC DRIVERS

Before defining datasources in JBoss EAP for your applications to use, you must first install the appropriate JDBC driver.

12.2.1. Installing a JDBC Driver as a Core Module

To install a JDBC driver as a core module, you must first [add the JDBC driver as a core module](#) and then [register the JDBC driver](#) in the **datasources** subsystem.

12.2.1.1. Add the JDBC Driver as a Core Module

JDBC drivers can be installed as a core module using the management CLI using the following steps.

1. Download the JDBC driver.

Download the appropriate JDBC driver from your database vendor. See [JDBC Driver Download Locations](#) for standard download locations for JDBC drivers of common databases.

Make sure to extract the archive if the JDBC driver JAR file is contained within a ZIP or TAR archive.

2. Start the JBoss EAP server.

3. Launch the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh
```

4. Use the **module add** management CLI command to add the new core module.

```
module add --name=MODULE_NAME --resources=PATH_TO_JDBC_JAR --
dependencies=DEPENDENCIES
```

For example, the following command adds a MySQL JDBC driver module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

Execute **module --help** for more details on using this command to add and remove modules.

You must next register it as a JDBC driver for it to be referenced by application datasources.

12.2.1.2. Register the JDBC Driver

After the driver has been [installed as a core module](#), you must register it as a JDBC driver using the following management CLI command. When running in a managed domain, be sure to precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME,driver-module-name=MODULE_NAME,driver-xa-datasource-class-name=XA_DATASOURCE_CLASS_NAME, driver-class-name=DRIVER_CLASS_NAME)
```

NOTE

The **driver-class-name** parameter is only required if the JDBC driver jar defines more than one class in its **/META-INF/services/java.sql.Driver** file.

For example, the **/META-INF/services/java.sql.Driver** file in the MySQL 5.1.36 JDBC driver JAR defines two classes:

- `com.mysql.cj.jdbc.Driver`
- `com.mysql.fabric.jdbc.FabricMySQLDriver`

For this case, you would pass in **driver-class-name=com.mysql.cj.jdbc.Driver**.

For example, the following command registers a MySQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.cj.jdbc.MySQLXADataSource, driver-class-name=com.mysql.cj.jdbc.Driver)
```

The JDBC driver is now available to be referenced by application datasources.

12.2.2. Installing a JDBC Driver as a JAR Deployment

JDBC drivers can be installed as a JAR deployment using either the management CLI or the management console. As long as the driver is JDBC 4-compliant, it will automatically be recognized and installed as a JDBC driver upon deployment.

The following steps describe how to install a JDBC driver using the management CLI.

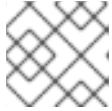
NOTE

The recommended installation method for JDBC drivers is to install them as a [core module](#).

1. Download the JDBC driver.
Download the appropriate JDBC driver from your database vendor. See [JDBC Driver Download Locations](#) for standard download locations for JDBC drivers of common databases.

Make sure to extract the archive if the JDBC driver JAR file is contained within a ZIP or TAR archive.
2. If the JDBC driver is not JDBC 4-compliant, see the steps to [Update a JDBC Driver JAR to be JDBC 4-Compliant](#).
3. Deploy the JAR to JBoss EAP.

```
deploy PATH_TO_JDBC_JAR
```

**NOTE**

In a managed domain, specify the appropriate server groups.

For example, the following command deploys a MySQL JDBC driver.

```
deploy /path/to/mysql-connector-java-8.0.12.jar
```

A message will be displayed in the JBoss EAP server log that displays the deployed driver name, which will be used when defining datasources.

```
WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-8.0.12.jar
```

The JDBC driver is now available to be referenced by application datasources.

Update a JDBC Driver JAR to be JDBC 4-Compliant

If the JDBC driver JAR is not JDBC 4-compliant, it can be made deployable using the following steps.

1. Create an empty temporary directory.
2. Create a **META-INF** subdirectory.
3. Create a **META-INF/services** subdirectory.
4. Create a **META-INF/services/java.sql.Driver** file and add one line to indicate the fully-qualified class name of the JDBC driver.

For example, the below line would be added for a MySQL JDBC driver.

```
com.mysql.cj.jdbc.Driver
```

5. Use the JAR command-line tool to add this new file to the JAR.

```
jar \-uf jdbc-driver.jar META-INF/services/java.sql.Driver
```

12.2.3. JDBC Driver Download Locations

The following table gives the standard download locations for JDBC drivers of common databases used with JBoss EAP.

**NOTE**

These links point to third-party websites which are not controlled or actively monitored by Red Hat. For the most up-to-date drivers for your database, check your database vendor's documentation and website.

Table 12.1. JDBC Driver Download Locations

Vendor	Download Location
MySQL	http://www.mysql.com/products/connector/
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
IBM	http://www-01.ibm.com/support/docview.wss?uid=swg21363866
Sybase	The jConnect JDBC driver is part of the SDK for SAP ASE installation. Currently there is not a separate download site for this driver by itself.
Microsoft	http://msdn.microsoft.com/data/jdbc/

12.2.4. Access Vendor-Specific Classes

In some cases, it is necessary for an application to use vendor-specific functionality that is not part of the JDBC API. In these cases, you can access vendor-specific APIs by declaring a dependency in that application.



WARNING

This is advanced usage. Only applications that need functionality not found in the JDBC API should implement this procedure.



IMPORTANT

This process is required when using the reauthentication mechanism, and accessing vendor-specific classes.

You can define a dependency for an application using either the **MANIFEST.MF** file or a **jboss-deployment-structure.xml** file.

If you have not yet done so, [install a JDBC driver as a core module](#).

Using the MANIFEST.MF File

1. Edit the application's **META-INF/MANIFEST.MF** file.
2. Add the **Dependencies** line and specify the module name.
For example, the below line declares the **com.mysql** module as a dependency.

```
Dependencies: com.mysql
```

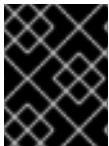

Using a `jboss-deployment-structure.xml` File

1. Create a file called **`jboss-deployment-structure.xml`** in the **`META-INF/`** or **`WEB-INF/`** folder of the application.
2. Specify the module using the **`dependencies`** element.
For example, the following example **`jboss-deployment-structure.xml`** file declares the **`com.mysql`** module as a dependency.

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

The example code below accesses the MySQL API.

```
import java.sql.Connection;
...
Connection c = ds.getConnection();
if (c.isWrapperFor(com.mysql.jdbc.Connection.class)) {
    com.mysql.jdbc.Connection mc = c.unwrap(com.mysql.jdbc.Connection.class);
}
```



IMPORTANT

Follow the vendor-specific API guidelines closely, as the connection is being controlled by the IronJacamar container.

12.3. CREATING DATASOURCES

Datasources can be created using the management console or the management CLI.

JBoss EAP 7 allows you to use expressions in datasource attribute values, such as the **`enabled`** attribute. See the [Property Replacement](#) section for details on using expressions in the configuration.

12.3.1. Create a Non-XA Datasource

You can create non-XA datasources using either the management CLI or the management console.

Defining a Non-XA Datasource Using the Management Console

1. Navigate to **Configuration → Subsystems → Datasources & Drivers → Datasources**.
2. Click the Add (+) button and choose **Add Datasource**.
3. It opens the **Add Datasource** wizard where you can choose the datasource type and click **Next**. This creates a template for your database. The following pages of the wizard are prefilled with values specific for the selected datasource. This makes the datasource creation process easy.

4. You can test your connection on the **Test Connection** page before finishing the datasource creation process.
5. Review the details and click **Finish** to create the datasource.

Defining a Non-XA Datasource Using the Management CLI

Non-XA datasources can be defined using the **data-source add** management CLI command.

1. If you have not yet done so, install and register the appropriate [JDBC Driver as a Core Module](#).
2. Define the datasource using the **data-source add** command, specifying the appropriate argument values.

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --connection-url=CONNECTION_URL
```



NOTE

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

See the [Datasource Parameters](#) section below for tips on these parameter values.

For detailed examples, see the [Example Datasource Configurations](#) for the supported databases.

Datasource Parameters

jndi-name

The JNDI name for the datasource must start with **java:/** or **java:jboss/**. For example, **java:jboss/datasources/ExampleDS**.

driver-name

The driver name value depends on whether the JDBC driver was installed as a core module or a JAR deployment.

1. For a core module, the driver name value will be the name given to the JDBC driver when it was registered.
2. For a JAR deployment, the driver name is the name of the JAR if there is only one class listed in its **/META-INF/services/java.sql.Driver** file. If there are multiple classes listed, then the value is **JAR_NAME** + "_" + **DRIVER_CLASS_NAME** + "_" + **MAJOR_VERSION** + "_" + **MINOR_VERSION** (for example **mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1**).

You can also see the driver name listed in the JBoss EAP server log when the JDBC JAR is deployed.

```
WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1
```

connection-url

For details on the connection URL formats for the supported databases, see the list of [Datasource Connection URLs](#).

For a complete listing of all available datasource attributes, see the [Datasource Attributes](#) section.

12.3.2. Create an XA Datasource

You can create XA datasources using either the management CLI or the management console.

Defining an XA Datasource Using the Management Console

1. Navigate to **Configuration → Subsystems → Datasources & Drivers → Datasources**.
2. Click the Add (+) button and choose **Add XA Datasource**.
3. It opens the **Add XA Datasource** wizard where you can choose the datasource type and click **Next**. This creates a template for your database. The following pages of the wizard are prefilled with values specific for the selected datasource. This makes the datasource creation process easy.
4. You can test your connection on the **Test Connection** page before finishing the datasource creation process.
5. Review the details and click **Finish** to create the datasource.

Defining an XA Datasource Using the Management CLI

XA datasources can be defined using the **xa-data-source add** management CLI command.



NOTE

In a managed domain, you will need to specify the profile to use. Depending on the format of the management CLI command, you will either precede the command with **/profile=PROFILE_NAME** or pass in the **--profile=PROFILE_NAME** argument.

1. If you have not yet done so, install and register the appropriate [JDBC Driver as a Core Module](#).
2. Define the datasource using the **xa-data-source add** command, specifying the appropriate argument values.

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --xa-datasource-class=XA_DATASOURCE_CLASS --xa-datasource-properties={"ServerName"=>"HOST_NAME","DatabaseName"=>"DATABASE_NAME"}
```

See the [Datasource Parameters](#) section below for tips on these parameter values.

3. Set *XA datasource properties*.

At least one *XA datasource property* is required when defining an XA datasource or you will receive an error when adding the datasource in the previous step. Any properties that were not set when defining the XA datasource can be set individually afterward.

- a. Set the server name.

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=ServerName:add(value=HOST_NAME)
```

- b. Set the database name.

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=DatabaseName:add(value=DATABASE_NAME)
```

For detailed examples, see the [Example Datasource Configurations](#) for the supported databases.

Datasource Parameters

jndi-name

The JNDI name for the datasource must start with **java:/** or **java:jboss/**. For example, **java:jboss/datasources/ExampleDS**.

driver-name

The driver name value depends on whether the JDBC driver was installed as a core module or a JAR deployment.

1. For a core module, the driver name value will be the name given to the JDBC driver when it was registered.
2. For a JAR deployment, the driver name is the name of the JAR if there is only one class listed in its **/META-INF/services/java.sql.Driver** file. If there are multiple classes listed, then the value is **JAR_NAME** + "_" + **DRIVER_CLASS_NAME** + "_" + **MAJOR_VERSION** + "_" + **MINOR_VERSION**, for example, **mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1**.

You can also see the driver name listed in the JBoss EAP server log when the JDBC JAR is deployed.

```
WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-5.1.36-
bin.jar_com.mysql.cj.jdbc.Driver_5_1
```

xa-datasource-class

Specify the XA datasource class for the JDBC driver's implementation of the **javax.sql.XADataSource** class.

xa-datasource-properties

At least one *XA datasource property* is required when defining an XA datasource or you will receive an error when attempting to add it. Additional properties can also be added to the XA datasource after it has been defined.

For a complete listing of all available datasource attributes, see the [Datasource Attributes](#) section.

12.4. MODIFYING DATASOURCES

Datasources settings can be configured using the management console or the management CLI.

JBoss EAP 7 allows you to use expressions in datasource attribute values, such as the **enabled** attribute. See the [Property Replacement](#) section for details on using expressions in the configuration.

12.4.1. Modify a Non-XA Datasource

Non-XA datasource settings can be updated using the **data-source** management CLI command. You can also update datasource attributes from the management console by navigating to **Configuration → Subsystems → Datasources & Drivers → Datasources**.



NOTE

Non-XA datasources can be integrated with JTA transactions. To integrate the datasource with JTA, ensure that the **jta** parameter is set to **true**.

Settings for a datasource can be updated by using the following management CLI command.

```
data-source --name=DATASOURCE_NAME --ATTRIBUTE_NAME=ATTRIBUTE_VALUE
```



NOTE

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

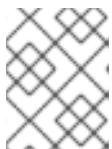
A server reload may be required for the changes to take effect.

12.4.2. Modify an XA Datasource

XA datasource settings can be updated using the **xa-data-source** management CLI command. You can also update datasource attributes from the management console by navigating to **Configuration → Subsystems → Datasources & Drivers → Datasources**.

- Settings for an XA datasource can be updated by using the following management CLI command.

```
xa-data-source --name=XA_DATASOURCE_NAME -  
-ATTRIBUTE_NAME=ATTRIBUTE_VALUE
```

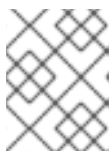


NOTE

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

- An *XA datasource property* can be added using the following management CLI command.

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-  
properties=PROPERTY:add(value=VALUE)
```



NOTE

In a managed domain, you must precede this command with **/profile=PROFILE_NAME**.

A server reload may be required for the changes to take effect.

12.5. REMOVING DATASOURCES

Datasources can be removed using the management console or the management CLI.

12.5.1. Remove a Non-XA Datasource

Non-XA datasources can be removed using the **data-source remove** management CLI command. You can also use the management console to remove datasources by navigating to **Configuration → Subsystems → Datasources & Drivers → Datasources**.

```
data-source remove --name=DATASOURCE_NAME
```

**NOTE**

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

The server will require a reload after the datasource is removed.

12.5.2. Remove an XA Datasource

XA datasources can be removed using the **xa-data-source remove** management CLI command. You can also use the management console to remove datasources by navigating to **Configuration → Subsystems → Datasources & Drivers → Datasources**.

```
xa-data-source remove --name=XA_DATASOURCE_NAME
```

**NOTE**

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

The server will require a reload after the XA datasource is removed.

12.6. TESTING DATASOURCE CONNECTIONS

You can use the management CLI or management console to test a datasource connection to verify that its settings are correct.

Test a Datasource Connection Using the Management CLI

The following management CLI command can be used to test a datasource's connection.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:test-connection-in-pool
```

**NOTE**

In a managed domain, you must precede this command with **/host=HOST_NAME/server=SERVER_NAME**. If you are testing an XA datasource, replace **data-source=DATASOURCE_NAME** with **xa-data-source=XA_DATASOURCE_NAME**.

Test a Datasource Connection Using the Management Console

When using the **Add Datasource** wizard in the management console, you have the opportunity to test the connection before creating the datasource. On the **Test Connection** screen of the wizard, click the **Test Connection** button.

Once a datasource has been added, you can test the connection by navigating to **Configuration → Subsystems → Datasources & Drivers → Datasources**, selecting the datasource, and choosing **Test Connection** from the drop down.

12.7. FLUSHING DATASOURCE CONNECTIONS

You can flush datasource connections using the following management CLI commands.



NOTE

In a managed domain, you must precede these commands with **/host=HOST_NAME/server=SERVER_NAME**.

- Flush all connections in the pool.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-all-connection-in-pool
```

- Gracefully flush all connections in the pool.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-gracefully-connection-in-pool
```

The server will wait until connections become idle before flushing them.

- Flush all idle connections in the pool.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-idle-connection-in-pool
```

- Flush all invalid connections in the pool.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-invalid-connection-in-pool
```

The server will flush all connections that it determines to be invalid, for example, by the **valid-connection-checker-class-name** or **check-valid-connection-sql** validation mechanism discussed in [Database Connection Validation](#).

You can also flush connections using the management console. From the **Runtime** tab, select the server, select **Datasources**, choose a datasource, and use the drop down to select the appropriate action.

12.8. XA DATASOURCE RECOVERY

An XA datasource is a datasource that can participate in an XA global transaction, which is coordinated by the transaction manager and can span multiple resources in a single transaction. If one of the participants fails to commit its changes, the other participants abort the transaction and restore their state as it was before the transaction occurred. This is to maintain consistency and avoid potential data loss or corruption.

XA recovery is the process of ensuring that all resources affected by a transaction are updated or rolled back, even if any of the resources or transaction participants crash or become unavailable. XA recovery happens without user intervention.

Each XA resource needs to have a recovery module associated with its configuration. The recovery module is the code that is executed when recovery is being performed. JBoss EAP automatically registers recovery modules for JDBC XA resources. You can register a custom module with your XA datasource if you wish to implement custom recovery code. The recovery module must extend class **com.arjuna.ats.jta.recovery.XAResourceRecovery**.

12.8.1. Configuring XA Recovery

For most JDBC resources, the recovery module is automatically associated with the resource. In these cases, you only need to configure the options that allow the recovery module to connect to your resource to perform recovery.

The following table describes the XA datasource parameters specific to XA recovery. Each of these configuration attributes can be set during datasource creation or afterward. You can set them using either the management console or the management CLI. See [Modify an XA Datasource](#) for information on configuring XA datasources.

Table 12.2. Datasource Parameters for XA Recovery

Attribute	Description
recovery-username	The user name to use to connect to the resource for recovery. Note that if this is not specified, the datasource security settings will be used.
recovery-password	The password to use to connect to the resource for recovery. Note that if this is not specified, the datasource security settings will be used.
recovery-security-domain	The security domain to use to connect to the resource for recovery.
recovery-plugin-class-name	If you need to use a custom recovery module, set this attribute to the fully-qualified class name of the module. The module should extend class com.arjuna.ats.jta.recovery.XAResourceRecovery .
recovery-plugin-properties	If you use a custom recovery module which requires properties to be set, set this attribute to the list of comma-separated KEY=VALUE pairs for the properties.

Disable XA Recovery

If multiple XA datasources connect to the same physical database, then XA recovery typically needs to be configured for only one of them.

Use the following management CLI command to disable recovery for an XA datasource:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-attribute(name=no-recovery,value=true)
```

12.8.2. Vendor-Specific XA Recovery

Vendor-Specific Configuration

Some databases require specific configurations in order to cooperate in XA transactions managed by the JBoss EAP transaction manager. For detailed and up-to-date information, see your database vendor's documentation.

MySQL

No special configuration is required. For more information, see the MySQL documentation.

PostgreSQL and Postgres Plus Advanced Server

For PostgreSQL to be able to handle XA transactions, change the configuration parameter **max_prepared_transactions** to a value greater than **0** and equal to or greater than **max_connections**.

Oracle

Ensure that the Oracle user, **USER**, has access to the tables needed for recovery.

```
GRANT SELECT ON sys.dba_pending_transactions TO USER;
GRANT SELECT ON sys.pending_trans$ TO USER;
GRANT SELECT ON sys.dba_2pc_pending TO USER;
GRANT EXECUTE ON sys.dbms_xa TO USER;
```

If the Oracle user does not have the proper permissions, you may see an error such as the following:

```
WARN [com.arjuna.ats.jta.logging.logger18N] [com.arjuna.ats.internal.jta.recovery.xarecovery1]
Local XARecoveryModule.xaRecovery got XA exception javax.transaction.xa.XAException,
XAException.XAER_RMERR
```

Microsoft SQL Server

For more information, see the Microsoft SQL Server documentation, including <http://msdn.microsoft.com/en-us/library/aa342335.aspx>.

IBM DB2

No special configuration is required. For more information, see the IBM DB2 documentation.

Sybase

Sybase expects XA transactions to be enabled on the database. Without correct database configuration, XA transactions will not work. The **enable xact coordination** parameter enables or disables Adaptive Server transaction coordination services. When this parameter is enabled, Adaptive Server ensures that updates to remote Adaptive Server data commit or roll back with the original transaction.

To enable transaction coordination, use:

```
sp_configure 'enable xact coordination', 1
```

MariaDB

No special configuration is required. For more information, see the MariaDB documentation.

Known Issues

These known issues with handling XA transactions are for the specific database and JDBC driver versions supported with JBoss EAP 7. For up-to-date information on the supported databases, see [JBoss EAP supported configurations](#).

MySQL

MySQL is not fully capable of handling XA transactions. If a client is disconnected from MySQL, then all the information about such transactions is lost. See this [MySQL bug](#) for more information. Note that this issue was fixed in MySQL 5.7.

PostgreSQL and Postgres Plus Advanced Server

The JDBC driver returns the **XAER_RMERR** XAException error code when a network failure occurs during the commit phase of two-phase commit (2PC). This error signals an unrecoverable catastrophic event to the transaction manager, but the transaction is left in the **in-doubt** state on the database side and could be easily corrected after network connection is established again. The correct return code should be **XAER_RMFAIL** or **XAER_RETRY**. The incorrect error code causes the transaction to be left in the **Heuristic** state on the JBoss EAP side and holding locks in the database which requires manual intervention. See this [PostgreSQL bug](#) for more information.

If a connection failure happens when the one-phase commit optimization is used, the JDBC driver returns **XAER_RMERR**, but the **XAER_RMFAIL** error code should be returned. This could lead to

data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

The Postgres Plus JDBC driver returns XIDs for all prepared transactions that exist in the Postgres Plus Server, so there is no way to determine the database to which the XID belongs. If you define more than one data source for the same database in JBoss EAP, an in-doubt transaction recovery attempt could be run under wrong account, which causes the recovery to fail.

Oracle

The JDBC driver returns XIDs belonging to all users on the database instance, when the Recovery Manager calls recovery using datasource configured with some user credentials. The JDBC driver throws the exception **ORA-24774: cannot switch to specified transaction** because it tries to recover XIDs belonging to other users.

The workaround for this issue is to grant **FORCE ANY TRANSACTION** privilege to the user whose credentials are used in recovery datasource configuration. More information about configuring the privilege can be found here:

http://docs.oracle.com/database/121/ADMIN/ds_txnman.htm#ADMIN12259.

Microsoft SQL Server

The JDBC driver returns the **XAER_RMERR** XAException error code when a network failure occurs during the commit phase of two-phase commit (2PC). This error signals an unrecoverable catastrophic event to the transaction manager, but the transaction is left in the **in-doubt** state on the database side and could be easily corrected after network connection is established again. The correct return code should be **XAER_RMFAIL** or **XAER_RETRY**. The incorrect error code causes the transaction to be left in the **Heuristic** state on the JBoss EAP side and holding locks in the database which requires manual intervention. See this [Microsoft SQL Server issue report](#) for more information.

If a connection failure happens when the one-phase commit optimization is used, the JDBC driver returns **XAER_RMERR**, but the **XAER_RMFAIL** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

IBM DB2

If a connection failure happens during a one-phase commit, the JDBC driver returns **XAER_RETRY**, but the **XAER_RMFAIL** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

Sybase

The JDBC driver returns the **XAER_RMERR** XAException error code when a network failure occurs during the commit phase of two-phase commit (2PC). This error signals an unrecoverable catastrophic event to the transaction manager, but the transaction is left in the **in-doubt** state on the database side and could be easily corrected after network connection is established again. The correct return code should be **XAER_RMFAIL** or **XAER_RETRY**. The incorrect error code causes the transaction to be left in the **Heuristic** state on the JBoss EAP side and holding locks in the database which requires manual intervention.

If a connection failure happens when the one-phase commit optimization is used, the JDBC driver returns **XAER_RMERR**, but the **XAER_RMFAIL** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

If an XA transaction that includes an insert to a Sybase 15.7 or 16 database fails before the Sybase transaction branch is in a prepared state, then repeating the XA transaction and inserting the same record with the same primary key will fail with an error of

com.sybase.jdbc4.jdbc.SybSQLException: Attempt to insert duplicate key row. This exception will be thrown until the original unfinished Sybase transaction branch is rolled back.

MariaDB

MariaDB is not fully capable of handling XA transactions. If a client is disconnected from MariaDB, then all the information about such transactions is lost.

MariaDB Galera Cluster

XA transactions are not supported in MariaDB Galera Cluster.

12.9. DATABASE CONNECTION VALIDATION

Database maintenance, network problems, or other outage events may cause JBoss EAP to lose the connection to the database. In order to recover from these situations, you can enable database connection validation for your datasources.

To configure database connection validation, you specify the validation timing method to define when the validation occurs, the validation mechanism to determine how the validation is performed, and the exception sorter to define how exceptions are handled.

1. Choose *one* of the validation timing methods.

validate-on-match

When the **validate-on-match** option is set to **true**, the database connection is validated every time it is checked out from the connection pool using the validation mechanism specified in the next step.

If a connection is not valid, a warning is written to the log and the next connection in the pool is retrieved. This process continues until a valid connection is found. If you prefer not to cycle through every connection in the pool, you can use the **use-fast-fail** option. If a valid connection is not found in the pool, a new connection is created. If the connection creation fails, an exception is returned to the requesting application.

This setting results in the quickest recovery but creates the highest load on the database. However, this is the safest selection if the minimal performance hit is not a concern.

background-validation

When the **background-validation** option is set to **true**, connections are validated periodically in a background thread prior to use. The frequency of the validation is specified by the **background-validation-millis** property. The default value of **background-validation-millis** is **0**, meaning that it is disabled.

When determining the value of the **background-validation-millis** property, consider the following:

- This value should not be set to the same value as your **idle-timeout-minutes** setting.
- The lower the value, the more frequently the pool is validated and the sooner invalid connections are removed from the pool.
- Lower values take more database resources. Higher values result in less frequent connection validation checks and use less database resources, but dead connections are undetected for longer periods of time.

**NOTE**

These options are mutually exclusive. If **validate-on-match** is set to **true**, then **background-validation** must be set to **false**. If **background-validation** is set to **true**, then **validate-on-match** must be set to **false**.

2. Choose *one* of the validation mechanisms.

valid-connection-checker-class-name

Using **valid-connection-checker-class-name** is the preferred validation mechanism. This specifies a connection checker class that is used to validate connections for the particular database in use. JBoss EAP provides the following connection checkers:

- **org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker**
- **org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker**

check-valid-connection-sql

Using **check-valid-connection-sql**, you provide the SQL statement that will be used to validate the connection.

The following is an example SQL statement that you might use to validate Oracle connections.

```
select 1 from dual
```

The following is an example SQL statement that you might use to validate MySQL or PostgreSQL connections.

```
select 1
```

3. Set the exception sorter class name.

When an exception is marked as fatal, the connection is closed immediately, even if the connection is participating in a transaction. Use the exception sorter class option to properly detect and clean up after fatal connection exceptions. Choose the appropriate JBoss EAP exception sorter for your datasource type.

- **org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter**

- `org.jboss.jca.adapters.jdbc.extensions.informix.InformixExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter`

12.10. DATASOURCE SECURITY

Datasource security refers to encrypting or obscuring passwords for datasource connections. These passwords can be stored in plain text in configuration files, but this represents a security risk.

There are several methods you can use for datasource security. Examples of each are included below.

Secure a Datasource Using a Security Domain

A security domain for the datasource is defined.

```
<security-domain name="DsRealm" cache-type="default">
  <authentication>
    <login-module code="ConfiguredIdentity" flag="required">
      <module-option name="userName" value="sa"/>
      <module-option name="principal" value="sa"/>
      <module-option name="password" value="sa"/>
    </login-module>
  </authentication>
</security-domain>
```



NOTE

If a security domain will be used with multiple datasources, then caching should be disabled on the security domain. This can be accomplished by setting the value of the **cache-type** attribute to **none** or by removing the attribute altogether; however, if caching is desired, then a separate security domain should be used for each datasource.

The **DsRealm** security domain is then referenced by the datasource configuration.

```
<datasources>
  <datasource jndi-name="java:jboss/datasources/securityDs"
    pool-name="securityDs">
    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
    <driver>h2</driver>
    <new-connection-sql>select current_user()</new-connection-sql>
    <security>
      <security-domain>DsRealm</security-domain>
    </security>
  </datasource>
</datasources>
```

For more information on using Security Domains, see [How to Configure Identity Management](#).

Secure a Datasource Using a Password Vault

```
<security>
  <user-name>admin</user-name>

  <password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmEtMW
  NIMDMYNDdmNmI2TEIORV9CUkVBS3ZhdWx0}</password>
</security>
```

For more information on using the Password Vault, see the [Password Vault](#) section in the JBoss EAP *How to Configure Server Security* guide.

Secure a Datasource Using a Credential Store

You can also use a credential store to provide passwords. The **elytron** subsystem provides the ability to create credential stores to securely house and use your passwords throughout JBoss EAP. You can find more details on creating and using credential stores in the [Credential Store](#) section in the JBoss EAP *How to Configure Server Security* guide.

Adding a Credential Store Reference to ExampleDS

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=credential-reference,value=
{store=exampleCS, alias=example-ds-pw})
```

Secure a Datasource Using an Authentication Context

You can also use an Elytron authentication context to provide usernames and passwords.

Use the following steps to configure and use an authentication context for datasource security.

1. Remove **password** and **user-name**.

```
/subsystem=datasources/data-source=ExampleDS:undefine-attribute(name=password)
/subsystem=datasources/data-source=ExampleDS:undefine-attribute(name=user-name)
```

2. Enable Elytron security for the datasource.

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=elytron-
enabled,value=true)

reload
```

3. Create an **authentication-configuration** for your credentials.

The authentication configuration contains the credentials you want your datasource to use when making a connection. The below example uses a reference to a credential store, but you could also use an Elytron security domain.

```
/subsystem=elytron/authentication-configuration=exampleAuthConfig:add(authentication-
name=sa,credential-reference={clear-text=sa})
```

4. Create an **authentication-context**.

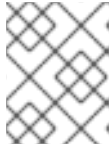
```
/subsystem=elytron/authentication-context=exampleAuthContext:add(match-rules=
[authentication-configuration=exampleAuthConfig])
```

5. Update the datasource to use the authentication context.

The below example updates **ExampleDS** to use an authentication context.

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=authentication-
context,value=exampleAuthContext)

reload
```



NOTE

If **authentication-context** is not set and **elytron-enabled** is set to **true**, JBoss EAP will use the current context for authentication.

Secure a Datasource using Kerberos

To secure a datasource using kerberos authentication, the following configuration is required:

- Kerberos is configured on the database server.
- JBoss EAP host server has a keytab entry for the database server.

To secure a datasource using kerberos:

1. Configure JBoss EAP to use kerberos.

```
/system-property=java.security.krb5.conf:add(value="/path/to/krb5.conf")
/system-property=sun.security.krb5.debug:add(value="false")
/system-property=sun.security.spnego.debug:add(value="false")
```

For debugging, change the values of **sun.security.krb5.debug** and **sun.security.spnego.debug** to **true**. In a production environment, it is recommended to set the values to **false**.

2. Configure security.

You can use legacy security or Elytron security to secure a datasource.

- To use kerberos with legacy security:
 - a. Configure infinispan cache to periodically remove expired tickets from cache.

```
batch
/subsystem=infinispan/cache-container=security:add(default-cache=auth-cache)
/subsystem=infinispan/cache-container=security/local-cache=auth-cache:add()
/subsystem=infinispan/cache-container=security/local-cache=auth-
cache/expiration=EXPIRATION:add(lifespan=3540000,max-idle=3540000)
/subsystem=infinispan/cache-container=security/local-cache=auth-
cache/memory=object:add(size=1000)
run-batch
```

The following attributes define ticket expiration:

- **lifespan**: Interval, in milliseconds, at which new certificates are requested from the KDC. Set the value of the **lifespan** attribute to be just smaller than the lifespan defined by the KDC.

- **max-idle**: Interval, in milliseconds, at which a valid ticket should be removed from the cache, if it unused.
- **max-entries**: The maximum number copies of the kerberos tickets to keep in cache. The value corresponds to the number of configured connections in your datasource.

b. Create a security domain.

```
batch
/subsystem=security/security-domain=KerberosDatabase:add(cache-type=infinispan)
/subsystem=security/security-domain=KerberosDatabase/authentication=classic:add
/subsystem=security/security-
domain=KerberosDatabase/authentication=classic/login-
module="KerberosDatabase-
Module":add(code="org.jboss.security.negotiation.KerberosLoginModule",module="org.
jboss.security.negotiation",flag=required, module-options={ "debug" => "false",
"storeKey" => "false", "useKeyTab" => "true", "keyTab" => "/path/to/eap.keytab",
"principal" => "PRINCIPAL@SERVER.COM", "doNotPrompt" => "true",
"refreshKrb5Config" => "true", "isInitiator" => "true", "addGSSCredential" => "true",
"credentialLifetime" => "-1"})
run-batch
```

- When using Microsoft JDBC driver for SQL server, add attribute and value **"wrapGSSCredential" => "true"** in **module-options**.
 - For debugging, change the value of the **debug** attribute in **module-options** to **true**.
- To use kerberos with Elytron:
 - a. Set up a kerberos factory in Elytron.

```
/subsystem=elytron/kerberos-security-factory=krbsf:add(debug=false,
principal=PRINCIPAL@SERVER.COM, path=/path/to/keytab, request-lifetime=-1,
obtain-kerberos-ticket=true, server=false)
```

- For debugging, add attribute and value **debug = true**.
For a list of supported attributes, see the [Kerberos Security Factory Attributes](#) section in the *How to Configure Server Security guide*.
- b. Create an authentication configuration to use the kerberos factory.

```
/subsystem=elytron/authentication-configuration=kerberos-conf:add(kerberos-
security-factory=krbsf)
```

c. Create an authentication context.

```
/subsystem=elytron/authentication-context=ds-context:add(match-rules=
[authentication-configuration=kerberos-conf])
```

3. Secure the datasource with kerberos.

- If you use legacy security:
 - a. Configure the datasource to use a security domain.

■


```
/subsystem=datasources/data-source=KerberosDS:add(connection-url="URL", min-
pool-size=0, max-pool-size=10, jndi-name="java:jboss/datasource/KerberosDS",
driver-name=<jdbc-driver>.jar, security-domain=KerberosDatabase, allow-multiple-
users=false, pool-prefill=false, pool-use-strict-min=false, idle-timeout-minutes=2)
```

- b. Configure vendor-specific connection properties.

```
/subsystem=datasources/data-source=KerberosDS/connection-properties=
<connection-property-name>:add(value="(<kerberos-value>"))
```

Example: connection properties for Oracle database.

```
/subsystem=datasources/data-source=KerberosDS/connection-
properties=oracle.net.authentication_services:add(value="(KERBEROS5)")
```

- If you use Elytron:

- a. Configure the datasource to use an authentication context.

```
/subsystem=datasources/data-source=KerberosDS:add(connection-url="URL", min-
pool-size=0, max-pool-size=10, jndi-name="java:jboss/datasource/KerberosDS",
driver-name=<jdbc-driver>.jar, elytron-enabled=true, authentication-context=ds-
context, allow-multiple-users=false, pool-prefill=false, pool-use-strict-min=false, idle-
timeout-minutes=2)
```

- b. Configure vendor-specific connection properties.

```
/subsystem=datasources/data-source=KerberosDS/connection-properties=
<connection-property-name>:add(value="(<kerberos-value>"))
```

Example: connection properties for Oracle database

```
/subsystem=datasources/data-source=KerberosDS/connection-
properties=oracle.net.authentication_services:add(value="(KERBEROS5)")
```

When using kerberos authentication, it is recommended to use the following attributes and values for the datasource:

- **pool-prefill=false**
- **pool-use-strict-min=false**
- **idle-timeout-minutes**

For a list of supported attributes, see [Datasource Attributes](#).

12.11. DATASOURCE STATISTICS

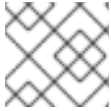
When statistics collection is [enabled](#) for a datasource, you can [view runtime statistics](#) for the datasource.

12.11.1. Enabling Datasource Statistics

By default, datasource statistics are *not* enabled. You can enable datasource statistics collection using the [management CLI](#) or the [management console](#).

Enable Datasource Statistics Using the Management CLI

The following management CLI command enables the collection of statistics for the **ExampleDS** datasource.



NOTE

In a managed domain, precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=statistics-
enabled,value=true)
```

Reload the server for the changes to take effect.

Enable Datasource Statistics Using the Management Console

Use the following steps to enable statistics collection for a datasource using the management console.

1. Navigate to **Configuration → Subsystems → Datasources & Drivers → Datasources**.
2. Select the datasource and click **View**.
3. Click **Edit** under the **Attributes** tab.
4. Set the **Statistics Enabled** field to **ON** and click **Save**. A popup appears indicating that the changes require a reload in order to take effect.
5. Reload the server.
 - For a standalone server, click the **Reload** link from the popup to reload the server.
 - For a managed domain, click the **Topology** link from the popup. From the **Topology** tab, select the appropriate server and select the **Reload** drop down option to reload the server.

12.11.2. Viewing Datasource Statistics

You can view runtime statistics for a datasource using the [management CLI](#) or [management console](#).

View Datasource Statistics Using the Management CLI

The following management CLI command retrieves the core *pool* statistics for the **ExampleDS** datasource.



NOTE

In a managed domain, precede these commands with **/host=HOST_NAME/server=SERVER_NAME**.

```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-
runtime=true)
{
  "outcome" => "success",
  "result" => {
    "ActiveCount" => 1,
```

```

    "AvailableCount" => 20,
    "AverageBlockingTime" => 0L,
    "AverageCreationTime" => 122L,
    "AverageGetTime" => 128L,
    "AveragePoolTime" => 0L,
    "AverageUsageTime" => 0L,
    "BlockingFailureCount" => 0,
    "CreatedCount" => 1,
    "DestroyedCount" => 0,
    "IdleCount" => 1,
    ...
}

```

The following management CLI command retrieves the *JDBC* statistics for the **ExampleDS** datasource.

```

/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-resource(include-
runtime=true)
{
  "outcome" => "success",
  "result" => {
    "PreparedStatementCacheAccessCount" => 0L,
    "PreparedStatementCacheAddCount" => 0L,
    "PreparedStatementCacheCurrentSize" => 0,
    "PreparedStatementCacheDeleteCount" => 0L,
    "PreparedStatementCacheHitCount" => 0L,
    "PreparedStatementCacheMissCount" => 0L,
    "statistics-enabled" => true
  }
}

```



NOTE

Since statistics are runtime information, be sure to specify the **include-runtime=true** argument.

See [Datasource Statistics](#) for a detailed list of all available statistics.

View Datasource Statistics Using the Management Console

To view datasource statistics from the management console, navigate to the **Datasources** subsystem from the **Runtime** tab, select a datasource, and click **View**.

See [Datasource Statistics](#) for a detailed list of all available statistics.

12.12. DATASOURCE TUNING

For tips on monitoring and optimizing performance for the **datasources** subsystem, see the [Datasource and Resource Adapter Tuning](#) section of the *Performance Tuning Guide*.

12.13. CAPACITY POLICIES

JBoss EAP supports defining capacity policies for Jakarta Connectors deployments, including datasources. Capacity policies define how physical connections for a pool are created, referred to as capacity incrementing, and destroyed, referred to as capacity decrementing. The default policies are set

to create one connection per request for capacity incrementing, and destroy all connections when they time out when the idle timeout is scheduled for capacity decrementing.

To configure capacity policies, you need to specify a capacity incrementer class, a capacity decrementer class, or both.

Example: Defining Capacity Policies

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-incrementer-class,
value="org.jboss.jca.core.connectionmanager.pool.capacity.SizeIncrementer")
```

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-decrementer-class,
value="org.jboss.jca.core.connectionmanager.pool.capacity.SizeDecrementer")
```

You can also configure properties on the specified capacity incrementer or decrementer class.

Example: Configuring Properties for Capacity Policies

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-incrementer-
properties.size, value=2)
```

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-decrementer-
properties.size, value=2)
```

MaxPoolSize Incrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.MaxPoolSizeIncrementer**

The *MaxPoolSize* incrementer policy will fill the pool to its max size for each request. This policy is useful when you want to keep the maximum number of connections available all the time.

Size Incrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.SizeIncrementer**

The *Size* incrementer policy will fill the pool by the specified number of connections for each request. This policy is useful when you want to increment with an additional number of connections per request in anticipation that the next request will also need a connection.

Table 12.3. Size policy properties

Name	Description
Size	The number of connections that should be created



NOTE

This is the default increment policy, and has a *size* value of 1.

Watermark Incrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.WatermarkIncrementer**

The *Watermark* incrementer policy will fill the pool to the specified number of connections for each request. This policy is useful when you want to keep a specified number of connections in the pool at all time.

Table 12.4. Watermark policy properties

Name	Description
Watermark	The watermark level for the number of connections

MinPoolSize Decrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.MinPoolSizeDecrementer**

The *MinPoolSize* decrementer policy will decrement the pool to its min size for each request. This policy is useful when you want to limit the number of connections after each idle timeout request. The pool will operate in a First In First Out (FIFO) manner.

Size Decrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.SizeDecrementer**

The *Size* decrementer policy will decrement the pool by the specified number of connections for each idle timeout request.

Table 12.5. Size policy properties

Name	Description
Size	The number of connections that should be destroyed

This policy is useful when you want to decrement an additional number of connections per idle timeout request in anticipation that the pool usage will lower over time.

The pool will operate in a First In First Out (FIFO) manner.

TimedOut Decrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.TimedOutDecrementer**

The *TimedOut* decrementer policy will removed all connections that have timed out from the pool for each idle timeout request. The pool will operate in a First In Last Out (FILO) manner.

**NOTE**

This policy is the default decrement policy.

TimedOut/FIFO Decrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.TimedOutFIFODecrementer**

The *TimedOutFIFO* decrementer policy will removed all connections that have timed out from the pool for each idle timeout request. The pool will operate in a First In First Out (FIFO) manner.

Watermark Decrementer Policy

Class name: **org.jboss.jca.core.connectionmanager.pool.capacity.WatermarkDecrementer**

The *Watermark* decrementer policy will decrement the pool to the specified number of connections for each idle timeout request. This policy is useful when you want to keep a specified number of connections in the pool at all time. The pool will operate in a First In First Out (FIFO) manner.

Table 12.6. Watermark policy properties

Name	Description
Watermark	The watermark level for the number of connections

12.14. ENLISTMENT TRACING

Enlistment traces can be recorded in order to help locate error situations that happen during enlistment of **XAResource** instances. When enlistment tracing is enabled, the **jca** subsystem creates an exception object for every pool operation, so that an accurate stack trace can be produced if necessary; however, this does come with performance overhead.

As of JBoss EAP 7.1, enlistment tracing is disabled by default. You can enable the recording of enlistment traces for a datasource using the management CLI by setting the **enlistment-trace** attribute to **true**.

Enable enlistment tracing for a non-XA datasource.

```
data-source --name=DATASOURCE_NAME --enlistment-trace=true
```

Enable enlistment tracing for an XA datasource.

```
xa-data-source --name=XA_DATASOURCE_NAME --enlistment-trace=true
```



WARNING

Enabling enlistment tracing can cause a performance impact.

12.15. EXAMPLE DATASOURCE CONFIGURATIONS

12.15.1. Example MySQL Datasource

This is an example of a MySQL datasource configuration with connection information, basic security, and validation options.

Example: MySQL Datasource Configuration

```
<datasources>
<datasource jndi-name="java:jboss/MySqlDS" pool-name="MySqlDS">
  <connection-url>jdbc:mysql://localhost:3306/jbossdb</connection-url>
  <driver>mysql</driver>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
```

```

    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
  </validation>
</datasource>
<drivers>
  <driver name="mysql" module="com.mysql">
    <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
    <xa-datasource-class>com.mysql.cj.jdbc.MySQLXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: MySQL JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MySQL JDBC driver as a core module.

```

module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api

```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MySQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-
name=com.mysql,driver-xa-datasource-class-name=com.mysql.cj.jdbc.MySQLXADataSource,
driver-class-name=com.mysql.cj.jdbc.Driver)
```

3. Add the MySQL datasource.

```
data-source add --name=MySqlDS --jndi-name=java:jboss/MySqlDS --driver-name=mysql --
connection-url=jdbc:mysql://localhost:3306/jbossdb --user-name=admin --password=admin --
validate-on-match=true --background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```

12.15.2. Example MySQL XA Datasource

This is an example of a MySQL XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: MySQL XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MySqlXADS" pool-name="MySqlXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mysql
    </xa-datasource-property>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
      <xa-datasource-class>com.mysql.cj.jdbc.MySQLXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example: MySQL JDBC Driver module.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
```



```
<resources>
  <resource-root path="mysql-connector-java-8.0.12.jar"/>
</resources>
<dependencies>
  <module name="javax.api"/>
  <module name="javax.transaction.api"/>
</dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MySQL JDBC driver as a core module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MySQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-
name=com.mysql,driver-xa-datasource-class-name=com.mysql.cj.jdbc.MySqlXADataSource,
driver-class-name=com.mysql.cj.jdbc.Driver)
```

3. Add the MySQL XA datasource.

```
xa-data-source add --name=MySqlXADS --jndi-name=java:jboss/MySqlXADS --driver-
name=mysql --user-name=admin --password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter --xa-
datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"mysqlpdb"}
```

12.15.3. Example PostgreSQL Datasource

This is an example of a PostgreSQL datasource configuration with connection information, basic security, and validation options.

Example: PostgreSQL Datasource Configuration

```

<datasources>
  <datasource jndi-name="java:jboss/PostgresDS" pool-name="PostgresDS">
    <connection-url>jdbc:postgresql://localhost:5432/postgresdb</connection-url>
    <driver>postgresql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
    </validation>
  </datasource>
</drivers>
  <driver name="postgresql" module="com.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: PostgreSQL JDBC Driver module.xml File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.postgresql">
  <resources>
    <resource-root path="postgresql-42.x.y.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

In the example above, make sure you replace 42.x.y with your driver version number.

Additional resources

- For information about JDBC drivers and how to install them, see [JDBC Drivers](#).
- For information about JDBC drivers that have been tested as part of the JBoss EAP 7.3, see [JBoss EAP 7.3 Supported Configurations](#).

Example Management CLI Commands

You can add the PostgreSQL JDBC driver and the PostgreSQL datasource to your JDBC API with the following CLI commands.

1. Add the PostgreSQL JDBC driver as a core module.

```
module add --name=com.postgresql --resources=/path/to/postgresql-42.x.y.jar --
dependencies=javax.api,javax.transaction.api
```

In the example above, make sure you replace 42.x.y with your driver version number.

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the PostgreSQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-name=postgresql,driver-module-
name=com.postgresql,driver-xa-datasource-class-
name=org.postgresql.xa.PGXADDataSource)
```

3. Add the PostgreSQL datasource.

```
data-source add --name=PostgresDS --jndi-name=java:jboss/PostgresDS --driver-
name=postgresql --connection-url=jdbc:postgresql://localhost:5432/postgresdb --user-
name=admin --password=admin --validate-on-match=true --background-validation=false --
valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker
--exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter
```

12.15.4. Example PostgreSQL XA Datasource

This is an example of a PostgreSQL XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: PostgreSQL XA Datasource Configuration

```
<datasources>
<xa-datasource jndi-name="java:jboss/PostgresXADS" pool-name="PostgresXADS">
  <xa-datasource-property name="ServerName">
    localhost
  </xa-datasource-property>
  <xa-datasource-property name="PortNumber">
    5432
  </xa-datasource-property>
  <xa-datasource-property name="DatabaseName">
    postgresdb
```

```

</xa-datasource-property>
<driver>postgresql</driver>
<security>
  <user-name>admin</user-name>
  <password>admin</password>
</security>
<validation>
  <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker"/>
  <validate-on-match>true</validate-on-match>
  <background-validation>false</background-validation>
  <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
</validation>
</xa-datasource>
<drivers>
  <driver name="postgresql" module="com.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: PostgreSQL JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.postgresql">
  <resources>
    <resource-root path="postgresql-42.x.y.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

In the example above, make sure you replace `42.x.y` with your driver version number.

Additional resources

- For information about JDBC drivers and how to install them, see [JDBC Drivers](#).
- For information about JDBC drivers that have been tested as part of the JBoss EAP 7.3, see [JBoss EAP 7.3 Supported Configurations](#).

Example Management CLI Commands

You can add the PostgreSQL JDBC driver and the PostgreSQL datasource to your JDBC API with the following CLI commands.

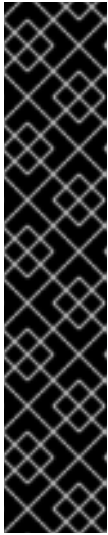
1. Add the PostgreSQL JDBC driver as a core module.

```

module add --name=com.postgresql --resources=/path/to/postgresql-42.x.y.jar --
dependencies=javax.api,javax.transaction.api

```

In the example above, make sure you replace `42.x.y` with your driver version number.



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the PostgreSQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-name=postgresql,driver-module-
name=com.postgresql,driver-xa-datasource-class-
name=org.postgresql.xa.PGXADDataSource)
```

3. Add the PostgreSQL XA datasource.

```
xa-data-source add --name=PostgresXADS --jndi-name=java:jboss/PostgresXADS --driver-
name=postgresql --user-name=admin --password=admin --validate-on-match=true --
background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker
--exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter --xa-
datasource-properties=
{"ServerName"=>"localhost","PortNumber"=>"5432","DatabaseName"=>"postgresdb"}
```

12.15.5. Example Oracle Datasource

This is an example of an Oracle datasource configuration with connection information, basic security, and validation options.

Example: Oracle Datasource Configuration

```
<datasources>
<datasource jndi-name="java:jboss/OracleDS" pool-name="OracleDS">
  <connection-url>jdbc:oracle:thin:@localhost:1521:XE</connection-url>
  <driver>oracle</driver>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
  </validation>
</datasource>
</datasources>
```

```

</validation>
</datasource>
<drivers>
  <driver name="oracle" module="com.oracle">
    <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: Oracle JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc7.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Oracle JDBC driver as a core module.

```

module add --name=com.oracle --resources=/path/to/ojdbc7.jar --
dependencies=javax.api,javax.transaction.api

```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the Oracle JDBC driver.

```

/subsystem=datasources/jdbc-driver=oracle:add(driver-name=oracle,driver-module-
name=com.oracle,driver-xa-datasource-class-
name=oracle.jdbc.xa.client.OracleXADataSource)

```

3. Add the Oracle datasource.

```
data-source add --name=OracleDS --jndi-name=java:jboss/OracleDS --driver-name=oracle -
-connection-url=jdbc:oracle:thin:@localhost:1521:XE --user-name=admin --password=admin
--validate-on-match=true --background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter
```

12.15.6. Example Oracle XA Datasource

IMPORTANT

The following settings must be applied for the user accessing an Oracle XA datasource in order for XA recovery to operate correctly. The value **user** is the user defined to connect from JBoss EAP to Oracle:

- **GRANT SELECT ON sys.dba_pending_transactions TO user;**
- **GRANT SELECT ON sys.pending_trans\$ TO user;**
- **GRANT SELECT ON sys.dba_2pc_pending TO user;**
- **GRANT EXECUTE ON sys.dbms_xa TO user;**

This is an example of an Oracle XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: Oracle XA Datasource Configuration

```
<datasources>
<xa-datasource jndi-name="java:jboss/OracleXADS" pool-name="OracleXADS">
  <xa-datasource-property name="URL">
    jdbc:oracle:thin:@oracleHostName:1521:orcl
  </xa-datasource-property>
  <driver>oracle</driver>
  <xa-pool>
    <is-same-rm-override>>false</is-same-rm-override>
  </xa-pool>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
  </validation>
</xa-datasource>
<drivers>
  <driver name="oracle" module="com.oracle">
    <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
```



```

</driver>
</drivers>
</datasources>

```

Example: Oracle JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc7.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Oracle JDBC driver as a core module.

```

module add --name=com.oracle --resources=/path/to/ojdbc7.jar --
dependencies=javax.api,javax.transaction.api

```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the Oracle JDBC driver.

```

/subsystem=datasources/jdbc-driver=oracle:add(driver-name=oracle,driver-module-
name=com.oracle,driver-xa-datasource-class-
name=oracle.jdbc.xa.client.OracleXADataSource)

```

3. Add the Oracle XA datasource.

```

xa-data-source add --name=OracleXADS --jndi-name=java:jboss/OracleXADS --driver-
name=oracle --user-name=admin --password=admin --validate-on-match=true --
background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker --
exception-sorter-class-

```



```
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter --same-rm-
override=false --xa-datasource-properties=
{"URL"=>"jdbc:oracle:thin:@oracleHostName:1521:orcl"}
```

12.15.7. Example Microsoft SQL Server Datasource

This is an example of a Microsoft SQL Server datasource configuration with connection information, basic security, and validation options.

Example: Microsoft SQL Server Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/MSSQLDS" pool-name="MSSQLDS">
    <connection-url>jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase</connection-url>
    <driver>sqlserver</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

Example: Microsoft SQL Server JDBC Drivermodule.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc42.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
    <module name="javax.xml.bind.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Microsoft SQL Server JDBC driver as a core module.

```
module add --name=com.microsoft --resources=/path/to/sqljdbc42.jar --
dependencies=javax.api,javax.transaction.api,javax.xml.bind.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the Microsoft SQL Server JDBC driver.

```
/subsystem=datasources/jdbc-driver=sqlserver:add(driver-name=sqlserver,driver-module-
name=com.microsoft,driver-xa-datasource-class-
name=com.microsoft.sqlserver.jdbc.SQLServerXADataSource)
```

3. Add the Microsoft SQL Server datasource.

```
data-source add --name=MSSQLDS --jndi-name=java:jboss/MSSQLDS --driver-
name=sqlserver --connection-
url=jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase --user-name=admin --
password=admin --validate-on-match=true --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSQLExceptionSorter
```

12.15.8. Example Microsoft SQL Server XA Datasource

This is an example of a Microsoft SQL Server XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: Microsoft SQL Server XA Datasource Configuration

```
<datasources>
<xa-datasource jndi-name="java:jboss/MSSQLXADS" pool-name="MSSQLXADS">
  <xa-datasource-property name="ServerName">
    localhost
  </xa-datasource-property>
  <xa-datasource-property name="DatabaseName">
    mssqldb
  </xa-datasource-property>
  <xa-datasource-property name="SelectMethod">
    cursor
  </xa-datasource-property>
</xa-datasource>
```

```

</xa-datasource-property>
<driver>sqlserver</driver>
<xa-pool>
  <is-same-rm-override>>false</is-same-rm-override>
</xa-pool>
<security>
  <user-name>admin</user-name>
  <password>admin</password>
</security>
<validation>
  <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"/>
  <validate-on-match>true</validate-on-match>
  <background-validation>>false</background-validation>
  <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSQLExceptionSorter"/>
</validation>
</xa-datasource>
<drivers>
  <driver name="sqlserver" module="com.microsoft">
    <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
class>
  </driver>
</drivers>
</datasources>

```

Example: Microsoft SQL Server JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc42.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
    <module name="javax.xml.bind.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

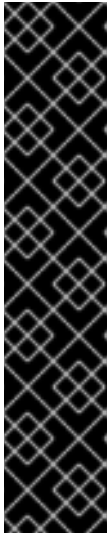
This example configuration can be achieved by using the following management CLI commands.

1. Add the Microsoft SQL Server JDBC driver as a core module.

```

module add --name=com.microsoft --resources=/path/to/sqljdbc42.jar --
dependencies=javax.api,javax.transaction.api,javax.xml.bind.api

```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the Microsoft SQL Server JDBC driver.

```
/subsystem=datasources/jdbc-driver=sqlserver:add(driver-name=sqlserver,driver-module-name=com.microsoft,driver-xa-datasource-class-name=com.microsoft.sqlserver.jdbc.SQLServerXADataSource)
```

3. Add the Microsoft SQL Server XA datasource.

```
xa-data-source add --name=MSSQLXADS --jndi-name=java:jboss/MSSQLXADS --driver-name=sqlserver --user-name=admin --password=admin --validate-on-match=true --background-validation=false --background-validation=disabled --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter --same-rm-override=false --xa-datasource-properties="{\"ServerName\"=>\"localhost\", \"DatabaseName\"=>\"mssqldb\", \"SelectMethod\"=>\"cursor\"}"
```

12.15.9. Example IBM DB2 Datasource

This is an example of an IBM DB2 datasource configuration with connection information, basic security, and validation options.

Example: IBM DB2 Datasource Configuration

```
<datasources>
<datasource jndi-name="java:jboss/DB2DS" pool-name="DB2DS">
  <connection-url>jdbc:db2://localhost:50000/ibmdb2db</connection-url>
  <driver>ibmdb2</driver>
  <pool>
    <min-pool-size>0</min-pool-size>
    <max-pool-size>50</max-pool-size>
  </pool>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"/>
  </validation>
</datasource>
</datasources>
```

```

    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"/>
  </validation>
</datasource>
<drivers>
  <driver name="ibmdb2" module="com.ibm">
    <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: IBM DB2 JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the IBM DB2 JDBC driver as a core module.

```

module add --name=com.ibm --resources=/path/to/db2jcc4.jar --
dependencies=javax.api,javax.transaction.api

```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the IBM DB2 JDBC driver.

```

/subsystem=datasources/jdbc-driver=ibmdb2:add(driver-name=ibmdb2,driver-module-
name=com.ibm,driver-xa-datasource-class-name=com.ibm.db2.jcc.DB2XADataSource)

```

3. Add the IBM DB2 datasource.

```
data-source add --name=DB2DS --jndi-name=java:jboss/DB2DS --driver-name=ibmdb2 --
connection-url=jdbc:db2://localhost:50000/ibmdb2db --user-name=admin --password=admin
--validate-on-match=true --background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker --exception-
sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter --min-
pool-size=0 --max-pool-size=50
```

12.15.10. Example IBM DB2 XA Datasource

This is an example of an IBM DB2 XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: IBM DB2 XA Datasource Configuration

```
<datasources>
<xa-datasource jndi-name="java:jboss/DB2XADS" pool-name="DB2XADS">
  <xa-datasource-property name="ServerName">
    localhost
  </xa-datasource-property>
  <xa-datasource-property name="DatabaseName">
    ibmdb2db
  </xa-datasource-property>
  <xa-datasource-property name="PortNumber">
    50000
  </xa-datasource-property>
  <xa-datasource-property name="DriverType">
    4
  </xa-datasource-property>
  <driver>ibmdb2</driver>
  <xa-pool>
    <is-same-rm-override>false</is-same-rm-override>
  </xa-pool>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <recovery>
    <recover-plugin class-name="org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin">
      <config-property name="EnableIsValid">
        false
      </config-property>
      <config-property name="IsValidOverride">
        false
      </config-property>
      <config-property name="EnableClose">
        false
      </config-property>
    </recover-plugin>
  </recovery>
  <validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"/>
    <validate-on-match>true</validate-on-match>
```

```

    <background-validation>false</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"/>
  </validation>
</xa-datasource>
<drivers>
  <driver name="ibmdb2" module="com.ibm">
    <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: IBM DB2 JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the IBM DB2 JDBC driver as a core module.

```

module add --name=com.ibm --resources=/path/to/db2jcc4.jar --
dependencies=javax.api,javax.transaction.api

```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the IBM DB2 JDBC driver.

```

/subsystem=datasources/jdbc-driver=ibmdb2:add(driver-name=ibmdb2,driver-module-
name=com.ibm,driver-xa-datasource-class-name=com.ibm.db2.jcc.DB2XADataSource)

```


3. Add the IBM DB2 XA datasource.

```
xa-data-source add --name=DB2XADS --jndi-name=java:jboss/DB2XADS --driver-
name=ibmdb2 --user-name=admin --password=admin --validate-on-match=true --
background-validation=false --background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker --exception-
sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter --same-
rm-override=false --recovery-plugin-class-
name=org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin --recovery-plugin-properties=
{"EnableIsValid"=>"false","IsValidOverride"=>"false","EnableClose"=>"false"} --xa-
datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"ibmdb2db","PortNumber"=>"50000","DriverT
ype"=>"4"}
```

12.15.11. Example Sybase Datasource

This is an example of a Sybase datasource configuration with connection information, basic security, and validation options.

Example: Sybase Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?
JCONNECT_VERSION=6</connection-url>
    <driver>sybase</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"/>
    </validation>
  </datasource>
</drivers>
  <driver name="sybase" module="com.sybase">
    <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>
```

Example: Sybase JDBC Driver module.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
```



```
<module name="javax.transaction.api"/>
</dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Sybase JDBC driver as a core module.

```
module add --name=com.sybase --resources=/path/to/jconn4.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the Sybase JDBC driver.

```
/subsystem=datasources/jdbc-driver=sybase:add(driver-name=sybase,driver-module-
name=com.sybase,driver-xa-datasource-class-
name=com.sybase.jdbc4.jdbc.SybXADataSource)
```

3. Add the Sybase datasource.

```
data-source add --name=SybaseDB --jndi-name=java:jboss/SybaseDB --driver-
name=sybase --connection-url=jdbc:sybase:Tds:localhost:5000/DATABASE?
JCONNECT_VERSION=6 --user-name=admin --password=admin --validate-on-match=true -
-background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter
```

12.15.12. Example Sybase XA Datasource

This is an example of a Sybase XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: Sybase XA Datasource Configuration

```
<datasources>
<xa-datasource jndi-name="java:jboss/SybaseXADS" pool-name="SybaseXADS">
<xa-datasource-property name="ServerName">
```

```

    localhost
  </xa-datasource-property>
  <xa-datasource-property name="DatabaseName">
    mydatabase
  </xa-datasource-property>
  <xa-datasource-property name="PortNumber">
    4100
  </xa-datasource-property>
  <xa-datasource-property name="NetworkProtocol">
    Tds
  </xa-datasource-property>
  <driver>sybase</driver>
  <xa-pool>
    <is-same-rm-override>false</is-same-rm-override>
  </xa-pool>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker"/>
    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"/>
  </validation>
</xa-datasource>
<drivers>
  <driver name="sybase" module="com.sybase">
    <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

Example: Sybase JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

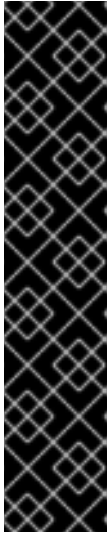
This example configuration can be achieved by using the following management CLI commands.

1. Add the Sybase JDBC driver as a core module.

```

module add --name=com.sybase --resources=/path/to/jconn4.jar --
dependencies=javax.api,javax.transaction.api

```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the Sybase JDBC driver.

```
/subsystem=datasources/jdbc-driver=sybase:add(driver-name=sybase,driver-module-
name=com.sybase,driver-xa-datasource-class-
name=com.sybase.jdbc4.jdbc.SybXADataSource)
```

3. Add the Sybase XA datasource.

```
xa-data-source add --name=SybaseXADS --jndi-name=java:jboss/SybaseXADS --driver-
name=sybase --user-name=admin --password=admin --validate-on-match=true --
background-validation=false --background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter --same-rm-
override=false --xa-datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"mydatabase","PortNumber"=>"4100","Netwo
rkProtocol"=>"Tds"}
```

12.15.13. Example MariaDB Datasource

This is an example of a MariaDB datasource configuration with connection information, basic security, and validation options.

Example: MariaDB Datasource Configuration

```
<datasources>
<datasource jndi-name="java:jboss/MariaDBDS" pool-name="MariaDBDS">
  <connection-url>jdbc:mariadb://localhost:3306/jbossdb</connection-url>
  <driver>mariadb</driver>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <exception-sorter class-
```

```

name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
</validation>
</datasource>
</drivers>
<driver name="mariadb" module="org.mariadb">
  <driver-class>org.mariadb.jdbc.Driver</driver-class>
  <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-datasource-class>
</driver>
</drivers>
</datasources>

```

Example: MariaDB JDBC Driver `module.xml` File

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.2.3.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MariaDB JDBC driver as a core module.

```

module add --name=org.mariadb --resources=/path/to/mariadb-java-client-1.2.3.jar --
dependencies=javax.api,javax.transaction.api

```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MariaDB JDBC driver.

```

/subsystem=datasources/jdbc-driver=mariadb:add(driver-name=mariadb,driver-module-
name=org.mariadb,driver-xa-datasource-class-name=org.mariadb.jdbc.MySQLDataSource,
driver-class-name=org.mariadb.jdbc.Driver)

```

3. Add the MariaDB datasource.

```
data-source add --name=MariaDBDS --jndi-name=java:jboss/MariaDBDS --driver-
name=mariadb --connection-url=jdbc:mariadb://localhost:3306/jbossdb --user-name=admin -
-password=admin --validate-on-match=true --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```

12.15.14. Example MariaDB XA Datasource

This is an example of a MariaDB XA datasource configuration with XA datasource properties, basic security, and validation options.

Example: MariaDB XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MariaDBXADS" pool-name="MariaDBXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mariadbdb
    </xa-datasource-property>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </xa-datasource>
</drivers>
<driver name="mariadb" module="org.mariadb">
  <driver-class>org.mariadb.jdbc.Driver</driver-class>
  <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-datasource-class>
</driver>
</drivers>
</datasources>
```

Example: MariaDB JDBC Driver module.xml File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.2.3.jar"/>
  </resources>
  <dependencies>
```

```
<module name="javax.api"/>
<module name="javax.transaction.api"/>
</dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MariaDB JDBC driver as a core module.

```
module add --name=org.mariadb --resources=/path/to/mariadb-java-client-1.2.3.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MariaDB JDBC driver.

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-name=mariadb,driver-module-
name=org.mariadb,driver-xa-datasource-class-name=org.mariadb.jdbc.MySQLDataSource,
driver-class-name=org.mariadb.jdbc.Driver)
```

3. Add the MariaDB XA datasource.

```
xa-data-source add --name=MariaDBXADS --jndi-name=java:jboss/MariaDBXADS --driver-
name=mariadb --user-name=admin --password=admin --validate-on-match=true --
background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter --xa-
datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"mariadbdb"}
```

12.15.15. Example MariaDB Galera Cluster Datasource

This is an example of a MariaDB Galera Cluster datasource configuration with connection information, basic security, and validation options.

**WARNING**

MariaDB Galera Cluster does not support XA transactions.

Example: MariaDB Galera Cluster Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/MariaDBGaleraClusterDS" pool-
name="MariaDBGaleraClusterDS">
    <connection-url>jdbc:mariadb://192.168.1.1:3306,192.168.1.2:3306/jbossdb</connection-url>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </datasource>
</drivers>
  <driver name="mariadb" module="org.mariadb">
    <driver-class>org.mariadb.jdbc.Driver</driver-class>
    <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>
```

Example: MariaDB JDBC Driver module.xml File

```
<?xml version='1.0' encoding='UTF-8'?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.5.4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MariaDB JDBC driver as a core module.


```
module add --name=org.mariadb --resources=/path/to/mariadb-java-client-1.5.4.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MariaDB JDBC driver.

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-name=mariadb,driver-module-
name=org.mariadb,driver-xa-datasource-class-name=org.mariadb.jdbc.MySQLDataSource,
driver-class-name=org.mariadb.jdbc.Driver)
```

3. Add the MariaDB Galera Cluster datasource.

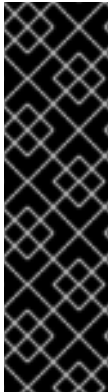
```
data-source add --name=MariaDBGaleraClusterDS --jndi-
name=java:jboss/MariaDBGaleraClusterDS --driver-name=mariadb --connection-
url=jdbc:mariadb://192.168.1.1:3306,192.168.1.2:3306/jbosssdb --user-name=admin --
password=admin --validate-on-match=true --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```


CHAPTER 13. DATASOURCE MANAGEMENT WITH AGROAL

13.1. ABOUT THE AGROAL DATASOURCES SUBSYSTEM

A new **datasources-agroal** subsystem was introduced in JBoss EAP 7.2. This is a new high performance direct connection pool backed by the Agroal project. This can be used as an alternative to the current JCA-based **datasources** subsystem.

The **datasources-agroal** subsystem is not available by default and you must [enable it](#) in order to use this new implementation.



IMPORTANT

The **datasources-agroal** subsystem is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

13.2. ENABLING THE AGROAL DATASOURCES SUBSYSTEM

The **datasources-agroal** subsystem is not enabled in the default JBoss EAP configuration. Use the following management CLI commands to enable it.

1. Add the extension.

```
/extension=org.wildfly.extension.datasources-agroal:add
```

2. Add the subsystem.

```
/subsystem=datasources-agroal:add
```

3. Reload the server for the changes to take effect.

```
reload
```

This adds the following XML in the server configuration file.

```
<server xmlns="urn:jboss:domain:8.0">
  <extensions>
    ...
    <extension module="org.wildfly.extension.datasources-agroal"/>
    ...
  </extensions>
  ...
  <subsystem xmlns="urn:jboss:domain:datasources-agroal:1.0"/>
  ...
</server>
```

13.3. INSTALLING A JDBC DRIVER AS A CORE MODULE FOR AGROAL DATASOURCES

Before defining Agroal datasources in JBoss EAP for your applications to use, you must first install the appropriate JDBC driver.

To install a JDBC driver as a core module for Agroal datasources, you must first [add the JDBC driver as a core module](#) and then [register the JDBC driver](#) in the **datasources-agroal** subsystem.

13.3.1. Add the JDBC Driver as a Core Module

JDBC drivers can be installed as a core module using the management CLI using the following steps.

1. Download the JDBC driver.

Download the appropriate JDBC driver from your database vendor. See [JDBC Driver Download Locations](#) for standard download locations for JDBC drivers of common databases.

Make sure to extract the archive if the JDBC driver JAR file is contained within a ZIP or TAR archive.

2. Start the JBoss EAP server.

3. Launch the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh
```

4. Use the **module add** management CLI command to add the new core module.

```
module add --name=MODULE_NAME --resources=PATH_TO_JDBC_JAR --
dependencies=DEPENDENCIES
```

For example, the following command adds a MySQL JDBC driver module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

Execute **module --help** for more details on using this command to add and remove modules.

You must next register it as a JDBC driver for it to be referenced by application datasources.

13.3.2. Register the JDBC Driver for Agroal Datasources

After the driver has been [installed as a core module](#), you must register it as a JDBC driver using the following management CLI command. When running in a managed domain, be sure to precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources-
agroal/driver=DRIVER_NAME:add(module=MODULE_NAME,class=CLASS_NAME)
```

The **CLASS_NAME** must be a fully qualified class name that implements either **java.sql.Driver** or **javax.sql.DataSource** for non-XA datasources, or **javax.sql.XADataSource** for XA datasources.

13.4. CONFIGURING AGROAL NON-XA DATASOURCES



IMPORTANT

The use of Agroal datasources is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

13.4.1. Create an Agroal Datasource

The following management CLI command creates an Agroal datasource. When running in a managed domain, be sure to precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources-agroal/datasource=DATASOURCE_NAME:add(jndi-
name=JNDI_NAME,connection-factory={driver=DRIVER_NAME,url=URL},connection-pool={max-
size=MAX_POOL_SIZE})
```

For a complete listing of all available Agroal datasource attributes, see the [Agroal Datasource Attributes](#) section.

See [Example MySQL Agroal Datasource](#) for an example Agroal datasource configuration.

13.4.2. Remove an Agroal Datasource

The following management CLI command removes an Agroal datasource. When running in a managed domain, be sure to precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources-agroal/datasource=DATASOURCE_NAME:remove
```

13.5. CONFIGURING AGROAL XA DATASOURCES



IMPORTANT

The use of Agroal datasources is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

13.5.1. Create an Agroal XA Datasource

The following management CLI command creates an Agroal XA datasource. When running in a managed domain, be sure to precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources-agroal/xa-datasource=XA_DATASOURCE_NAME:add(jndi-name=JNDI_NAME,connection-factory={driver=DRIVER_NAME,connection-properties={ServerName=HOST_NAME,PortNumber=PORT,DatabaseName=DATABASE_NAME}},connection-pool={max-size=MAX_POOL_SIZE})
```

For a complete listing of all available Agroal datasource attributes, see the [Agroal Datasource Attributes](#) section.

See [Example MySQL Agroal XA Datasource](#) for an example Agroal XA datasource configuration.

13.5.2. Remove an Agroal XA Datasource

The following management CLI command removes an Agroal XA datasource. When running in a managed domain, be sure to precede this command with **/profile=PROFILE_NAME**.

```
/subsystem=datasources-agroal/xa-datasource=DATASOURCE_NAME:remove
```

13.6. EXAMPLE AGROAL DATASOURCE CONFIGURATIONS

13.6.1. Example MySQL Agroal Datasource

This is an example of a MySQL Agroal datasource configuration with connection and basic security settings.

Example: MySQL Agroal Datasource Configuration

```
<subsystem xmlns="urn:jboss:domain:datasources-agroal:1.0">
  <datasource name="ExampleAgroalDS" jndi-name="java:jboss/datasources/ExampleAgroalDS">
    <connection-factory driver="mysql" url="jdbc:mysql://localhost:3306/jbossdb" username="admin"
password="admin"/>
    <connection-pool max-size="30"/>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql" class="com.mysql.cj.jdbc.Driver"/>
  </drivers>
</subsystem>
```

Example: MySQL JDBC Driver `module.xml` File

```
<?xml version='1.0' encoding='UTF-8'?>

<module xmlns="urn:jboss:module:1.1" name="com.mysql">

  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>

  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MySQL JDBC driver as a core module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MySQL JDBC driver.

```
/subsystem=datasources-
agroal/driver=mysql:add(module=com.mysql,class=com.mysql.cj.jdbc.Driver)
```

3. Add the MySQL datasource.

```
/subsystem=datasources-agroal/datasource=ExampleAgroalDS:add(jndi-
name=java:jboss/datasources/ExampleAgroalDS,connection-factory=
{driver=mysql,url=jdbc:mysql://localhost:3306/jbossdb,username=admin,password=admin},conn
ection-pool={max-size=30})
```

13.6.2. Example MySQL Agroal XA Datasource

This is an example of a MySQL Agroal XA datasource configuration with XA datasource properties and basic security settings.

Example: MySQL Agroal XA Datasource Configuration

```
<subsystem xmlns="urn:jboss:domain:datasources-agroal:1.0">
  <xa-datasource name="ExampleAgroalXADS" jndi-
name="java:jboss/datasources/ExampleAgroalXADS">
    <connection-factory driver="mysqlXA" username="admin" password="admin">
      <connection-properties>
        <property name="ServerName" value="localhost"/>
        <property name="PortNumber" value="3306"/>
        <property name="DatabaseName" value="jbossdb"/>
      </connection-properties>
    </connection-factory>
    <connection-pool max-size="30"/>
  </xa-datasource>
  <drivers>
    <driver name="mysqlXA" module="com.mysql" class="com.mysql.cj.jdbc.MysqlXADataSource"/>
  </drivers>
</subsystem>
```

Example: MySQL JDBC Driver `module.xml` File

```
<?xml version='1.0' encoding='UTF-8'?>

<module xmlns="urn:jboss:module:1.1" name="com.mysql">

  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>

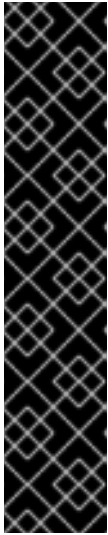
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MySQL JDBC driver as a core module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as Technology Preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be [added](#) and [removed](#) manually in a production environment.

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

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

2. Register the MySQL XA JDBC driver.

```
/subsystem=datasources-  
agroal/driver=mysqlXA:add(module=com.mysql,class=com.mysql.cj.jdbc.MysqlXADataSource)
```

3. Add the MySQL XA datasource.

```
/subsystem=datasources-agroal/xa-datasource=ExampleAgroalXADS:add(jndi-  
name=java:jboss/datasources/ExampleAgroalXADS,connection-factory=  
{driver=mysqlXA,connection-properties=  
{ServerName=localhost,PortNumber=3306,DatabaseName=jbossdb},username=admin,passwo  
rd=admin},connection-pool={max-size=30})
```

CHAPTER 14. CONFIGURING THE TRANSACTIONS SUBSYSTEM

The **transactions** subsystem allows you to configure the Transaction Manager (TM) options, such as timeout values, transaction logging, statistics collection, and whether to use JTS. JBoss EAP provides transactional services using the Narayana framework. This framework leverages support for a broad range of transaction protocols based on standards, such as Jakarta Transactions, JTS, and Web Service transactions.

For more information, see [Managing Transactions on JBoss EAP](#).

CHAPTER 15. ORB CONFIGURATION

15.1. ABOUT COMMON OBJECT REQUEST BROKER ARCHITECTURE (CORBA)

Common Object Request Broker Architecture (CORBA) is a standard that enables applications and services to work together even when they are written in multiple, otherwise-incompatible, languages or hosted on separate platforms. CORBA requests are brokered by a server-side component called an Object Request Broker (ORB). JBoss EAP provides an ORB instance, by means of the Open JDK ORB component.

The ORB is used internally for JTS transactions, and is also available for use by your own applications.



NOTE

The Object Transaction Service (OTS) is a cross-platform service that forms part of CORBA. The OTS specification is maintained by the Object Management Group. JTS is a specification for building a transaction manager, and JTS was designed based on the OTS specification.

For information about CORBA and its components, see [Common Object Request Broker Architecture](#).

15.2. CONFIGURING THE ORB FOR JTS

In a default installation of JBoss EAP, the Object Request Broker (ORB) support for transactions is disabled. You can configure ORB settings in the **iiop-openjdk** subsystem using the management CLI or the management console.



NOTE

The **iiop-openjdk** subsystem is available when using the *full* or *full-ha* profile in a managed domain, or the **standalone-full.xml** or **standalone-full-ha.xml** configuration file for a standalone server.

For a listing of the available configuration options for the **iiop-openjdk** subsystem, see [IIOP Subsystem Attributes](#).

Configure the ORB Using the Management CLI

You can configure each aspect of the ORB using the management CLI. This is the minimum configuration for the ORB to be used with JTS.

You can configure the following management CLI commands for a managed domain using the **full** profile. If necessary, change the profile to suit the one you need to configure. If you are using a standalone server, omit the **/profile=full** portion of the commands.

Enable the Security Interceptors

Enable the **security** attribute by setting the value to **identity**.

```
/profile=full/subsystem=iiop-openjdk:write-attribute(name=security,value=identity)
```

Enable Transactions in the IIOP Subsystem

To enable the ORB for JTS, set the value of **transactions** attribute to **full**, rather than the default **spec**.

```
/profile=full/subsystem=iiop-openjdk:write-attribute(name=transactions, value=full)
```

Enable JTS in the Transactions Subsystem

```
/profile=full/subsystem=transactions:write-attribute(name=jts,value=true)
```



NOTE

For JTS activation, the server must be restarted as reload is not enough.

Configure the ORB Using the Management Console

1. Select the **Configuration** tab from the top of the management console. In a managed domain, you must select the appropriate profile to modify.
2. Select **Subsystems** → **IIOP (OpenJDK)** and click **View**.
3. Click **Edit** and modify the attributes as needed.
4. Click **Save** to save the changes.

15.3. CONFIGURE IIOP TO USE SSL/TLS WITH THE ELYTRON SUBSYSTEM

You can configure the **iiop-openjdk** subsystem to use SSL/TLS to secure communication between clients and servers. The **elytron** subsystem, as well as the legacy **security** subsystem, provide the necessary components for configuring SSL/TLS for the **iiop-openjdk** subsystem as well as other subsystems within JBoss EAP. Use the following steps to configure the **iiop-openjdk** subsystem to use the **elytron** subsystem for SSL/TLS.

1. Use the following management CLI command to display the current legacy SSL/TLS configuration in the **iiop-openjdk** subsystem.

```
/subsystem=iiop-openjdk:read-attribute(name=security-domain)
{
  "outcome" => "success",
  "result" => "iiopSSLSecurityDomain"
}
```

The **iiop-openjdk** subsystem must use either the legacy **security** subsystem or the **elytron** subsystem for SSL/TLS. You cannot use both at the same time. The above command shows the **iiop-openjdk** subsystem is using a legacy security domain for handling SSL/TLS. Before you can configure the **iiop-openjdk** subsystem to use the **elytron** subsystem for SSL/TLS, you need to remove this reference:

```
/subsystem=iiop-openjdk:undefine-attribute(name=security-realm)
```

If the **security-domain** attribute in the **iiop-openjdk** is not defined, you can proceed to the next step.

2. Create a **server-ssl-context**.
To use SSL/TLS with the **iiop-openjdk** subsystem, you need to define a **server-ssl-context**. JBoss EAP uses the configuration provided by the **server-ssl-context** when making an

SSL/TLS connection as a server. You can find more details on creating a **server-ssl-context** in [Enable One-way SSL/TLS for Applications using the Elytron Subsystem](#) in *How to Configure Server Security* guide.

3. Create a **client-ssl-context**.

To use SSL/TLS with the **iiop-openjdk** subsystem, you need to define a **client-ssl-context**. JBoss EAP uses the configuration provided by the **client-ssl-context** when making an SSL/TLS connection as a client. You can find more details on creating a **client-ssl-context** in [Using a client-ssl-context](#) in the *How to Configure Server Security* guide.

4. Configure the **iiop-openjdk** subsystem to use the **client-ssl-context** and **server-ssl-context**.

Example: Setting client-ssl-context and server-ssl-context

```
batch

/subsystem=iiop-openjdk:write-attribute(name=client-ssl-context,value=iiopClientSSC)

/subsystem=iiop-openjdk:write-attribute(name=server-ssl-context,value=iiopServerSSC)

run-batch

reload
```

5. Configure the connection to and from the **iiop-openjdk** subsystem.

You can indicate whether or not SSL/TLS connections are required when connecting to and from the **iiop-openjdk** subsystem by adjusting the following attributes:

- To enable support for SSL in the **iiop-openjdk** subsystem, set **support-ssl** to **true**. Defaults to **false**.
- To require SSL/TLS connections from the **iiop-openjdk** subsystem, set **client-requires-ssl** to **true**. Defaults to **false**.
- To require SSL/TLS connections to the **iiop-openjdk** subsystem, set **server-requires-ssl** to **true**. Defaults to **false**. Note that setting this to **true** will block attempts to connect to the non-SSL IIOP socket.
- To adjust the **socket-binding**, set **ssl-socket-binding** to the desired binding. Defaults to **iiop-ssl**.

Example: Setting SSL/TLS Connections to and from IIOP as Required

```
/subsystem=iiop-openjdk:write-attribute(name=support-ssl,value=true)

/subsystem=iiop-openjdk:write-attribute(name=client-requires-ssl,value=true)

/subsystem=iiop-openjdk:write-attribute(name=server-requires-ssl,value=true)

/subsystem=iiop-openjdk:write-attribute(name=ssl-socket-binding,value=iiop-ssl)

reload
```

CHAPTER 16. JAKARTA CONNECTORS MANAGEMENT

16.1. ABOUT JAKARTA CONNECTORS

Jakarta Connectors define a standard architecture for Jakarta EE systems to external heterogeneous Enterprise Information Systems (EIS). Examples of EISs include Enterprise Resource Planning (ERP) systems, mainframe transaction processing (TP), databases, and messaging systems. A [resource adapter](#) is a component that implements the Jakarta Connectors architecture.

Jakarta Connectors 1.7 provides features for managing:

- connections
- transactions
- security
- life-cycle
- work instances
- transaction inflow
- message inflow

16.2. ABOUT RESOURCE ADAPTERS

A resource adapter is a deployable Jakarta EE component that provides communication between a Jakarta EE application and an Enterprise Information System (EIS) using the Jakarta Connectors specification. A resource adapter is often provided by EIS vendors to allow easy integration of their products with Jakarta EE applications.

An Enterprise Information System can be any other software system within an organization. Examples include Enterprise Resource Planning (ERP) systems, database systems, e-mail servers and proprietary messaging systems.

A resource adapter is packaged in a Resource Adapter Archive (RAR) file which can be deployed to JBoss EAP. A RAR file may also be included in an Enterprise Archive (EAR) deployment.

The resource adapter itself is defined within the **resource-adapters** subsystem, which is provided by the IronJacamar project.

16.3. CONFIGURING THE **jca** SUBSYSTEM

The **jca** subsystem controls the general settings for the Jakarta Connectors container and resource adapter deployments. You can configure the **jca** subsystem using the management console or the management CLI.

The main **jca** subsystem elements to configure are:

- [Archive validation](#)
- [Bean validation](#)
- [Work managers](#)

- [Distributed work managers](#)
- [Bootstrap contexts](#)
- [Cached connection manager](#)

Configure **jca** subsystem settings from the management console

The **jca** subsystem can be configured from the management console by navigating to **Configuration** → **Subsystems** → **JCA** and clicking **View**. Then, select the appropriate tab:

- **Configuration** - Contains settings for the cached connection manager, archive validation, and bean validation. Each of these is contained in their own tab as well. Modify these settings by opening the appropriate tab and clicking the **Edit** link.
- **Bootstrap Context** - Contains the list of configured bootstrap contexts. New bootstrap context objects can be added, removed, and configured. Each bootstrap context must be assigned a work manager.
- **Workmanager** - Contains the list of configured work managers. New work managers can be added, removed, and their thread pools configured here. Each work manager can have one short-running thread pool and an optional long-running thread pool.
The thread pool attributes can be configured by clicking **Thread Pools** on the selected work manager.

Configure **jca** subsystem settings from the management CLI

The **jca** subsystem can be configured from the management CLI from the `/subsystem=jca` address. In a managed domain, you must precede the command with `/profile=PROFILE_NAME`.



NOTE

Attribute names in the tables in the following sections are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at `EAP_HOME/docs/schema/wildfly-jca_5_0.xsd` to view the elements as they appear in the XML, as there may be differences from the management model.

Archive Validation

This determines whether archive validation will be performed on the deployment units. The following table describes the attributes you can set for archive validation.

Table 16.1. Archive Validation Attributes

Attribute	Default Value	Description
enabled	true	Specifies whether archive validation is enabled.
fail-on-error	true	Specifies whether an archive validation error report fails the deployment.
fail-on-warn	false	Specifies whether an archive validation warning report fails the deployment.

If an archive does not implement the Jakarta Connectors specification correctly and archive validation is enabled, an error message will display during deployment describing the problem. For example:

Severity: ERROR

Section: 19.4.2

Description: A ResourceAdapter must implement a "public int hashCode()" method.

Code: com.mycompany.myproject.ResourceAdapterImpl

Severity: ERROR

Section: 19.4.2

Description: A ResourceAdapter must implement a "public boolean equals(Object)" method.

Code: com.mycompany.myproject.ResourceAdapterImpl

If archive validation is not specified, it is considered present and the **enabled** attribute defaults to **true**.

Bean Validation

This setting determines whether bean validation, defined in [JSR-303](#), will be performed on the deployment units. The following table describes the attributes you can set for bean validation. The Jakarta equivalent for bean validation is [Jakarta Bean Validation](#).

Table 16.2. Bean Validation Attributes

Attribute	Default Value	Description
enabled	true	Specifies whether bean validation is enabled.

If bean validation is not specified, it is considered present and the **enabled** attribute defaults to **true**.

Work Managers

There are two types of work managers:

Default work manager

The default work manager and its thread pools.

Custom work manager

A custom work manager definition and its thread pools.

The following table describes the attributes you can set for work managers.

Table 16.3. Work Manager Attributes

Attribute	Description
name	Specifies the name of the work manager.
elytron-enabled	This attribute enables Elytron security for the workmanager .

A work manager also has the following child elements.

Table 16.4. Work Manager Child Elements

Child Element	Description
short-running-threads	Thread pool for standard Work instances. Each work manager has one short-running thread pool.
long-running-threads	Thread pool for Jakarta Connectors 1.7 Work instances that set the LONG_RUNNING hint. Each work manager can have one optional long-running thread pool.

The following table describes the attributes you can set for work manager thread pools.

Table 16.5. Thread Pool Attributes

Attribute	Description
allow-core-timeout	Boolean setting that determines whether core threads may time out. The default value is false .
core-threads	The core thread pool size. This must be equal to or smaller than the maximum thread pool size.
handoff-executor	An executor to delegate tasks to in the event that a task cannot be accepted. If not specified, tasks that cannot be accepted will be silently discarded.
keepalive-time	Specifies the amount of time that pool threads should be kept after doing work.
max-threads	The maximum thread pool size.
name	Specifies the name of the thread pool.
queue-length	The maximum queue length.
thread-factory	Reference to the thread factory.

Distributed Work Managers

A distributed work manager is a work manager instance that is able to reschedule work execution on another work manager instance.

The following example management CLI commands configure a distributed work manager. Note that you must use a configuration that provides high availability capabilities, such as the **standalone-ha.xml** or **standalone-full-ha.xml** configuration file for a standalone server.

Example: Configure a Distributed Work Manager

```
batch
/subsystem=jca/distributed-workmanager=myDistWorkMgr:add(name=myDistWorkMgr)
/subsystem=jca/distributed-workmanager=myDistWorkMgr/short-running-
threads=myDistWorkMgr:add(queue-length=10,max-threads=10)
```

```
/subsystem=jca/bootstrap-
context=myCustomContext:add(name=myCustomContext,workmanager=myDistWorkMgr)
run-batch
```



NOTE

The name of the **short-running-threads** element must be the same as the name of the **distributed-workmanager** element.

The following table describes the attributes you can configure for distributed work managers.

Table 16.6. Distributed Work Manager Attributes

Attribute	Description
elytron-enabled	Enables Elytron security for the work manager.
name	The name of the distributed work manager.
policy	<p>The policy decides when to redistribute a work instance. Allowed values are:</p> <ul style="list-style-type: none"> • NEVER - Never distribute the work instance to another node. • ALWAYS - Always distribute the work instance to another node. • WATERMARK - Distribute the work instance to another node based on how many free worker threads the current node has available.
policy-options	<p>List of the policy's key/value pair options. If you use the WATERMARK policy, then you can use the watermark policy option to specify at what number of free threads that work should be distributed. For example:</p> <pre>/subsystem=jca/distributed-workmanager=myDistWorkMgr:write-attribute(name=policy-options,value={watermark=3})</pre>
selector	<p>The selector decides to which nodes in the network to redistribute the work instance. Allowed values are:</p> <ul style="list-style-type: none"> • FIRST_AVAILABLE - Select the first available node in the list. • PING_TIME - Select the node with the lowest ping time. • MAX_FREE_THREADS - Select the node with highest number of free worker threads.
selector-options	List of the selector's key/value pair options.

A distributed work manager also has the following child elements.

Table 16.7. Distributed Work Manager Child Elements

Child Element	Description
long-running-threads	The thread pool for work instances that set the LONG_RUNNING hint. Each distributed work manager can optionally have a long-running thread pool.
short-running-threads	The thread pool for standard work instances. Each distributed work manager must have a short-running thread pool.

Bootstrap Contexts

This is used to define custom bootstrap contexts. The following table describes the attributes you can set for bootstrap contexts.

Table 16.8. Bootstrap Context Attributes

Attribute	Description
name	Specifies the name of the bootstrap context.
workmanager	Specifies the name of the work manager to use for this context.

Cached Connection Manager

This is used for debugging connections and supporting lazy enlistment of a connection in a transaction, tracking whether they are used and released properly by the application. The following table describes the attributes you can set for the cached connection manager.

Table 16.9. Cached Connection Manager Attributes

Attribute	Default Value	Description
debug	false	Outputs warning on failure to explicitly close connections.
error	false	Throws exception on failure to explicitly close connections.
ignore-unknown-connections	false	Specifies that unknown connections will not be cached.
install	false	Enable or disable the cached connection manager valve and interceptor.

16.4. CONFIGURING RESOURCE ADAPTERS

16.4.1. Deploy a Resource Adapter

Resource adapters can be deployed just like other deployments using the management CLI or the management console. When running a standalone server, you can also copy the archive to the deployments directory to be picked up by the deployment scanner.

Deploy a Resource Adapter using the Management CLI

To deploy the resource adapter to a standalone server, enter the following management CLI command.

```
deploy /path/to/resource-adapter.rar
```

To deploy the resource adapter to all server groups in a managed domain, enter the following management CLI command.

```
deploy /path/to/resource-adapter.rar --all-server-groups
```

Deploy a Resource Adapter using the Management Console

1. Log in to the management console and click on the **Deployments** tab.
2. Click the Add (+) button. In a managed domain, you will first need to select **Content Repository**.
3. Choose the **Upload Deployment** option.
4. Browse to the resource adapter archive and click **Next**.
5. Verify the upload, then click **Finish**.
6. In a managed domain, deploy the deployment to the appropriate server groups and enable the deployment.

Deploy a Resource Adapter Using the Deployment Scanner

To deploy a resource adapter manually to a standalone server, copy the resource adapter archive to the server deployments directory, for example, **EAP_HOME/standalone/deployments/**. This will be picked up and deployed by the deployment scanner.



NOTE

This option is not available for managed domains. You must use either the management console or the management CLI to deploy the resource adapter to server groups.

16.4.2. Configure a Resource Adapter

You can configure resource adapters using the management interfaces. The below example shows how to configure a resource adapter using the management CLI. See your resource adapter vendor's documentation for supported properties and other important information.

Add the Resource Adapter Configuration

Add the resource adapter configuration.

```
/subsystem=resource-adapters/resource-adapter=eis.rar:add(archive=eis.rar, transaction-  
support=XATransaction)
```

Configure the Resource Adapter Settings

Configure any of the following settings as necessary.

- Configure **config-properties**.
Add the **server** configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/config-  
properties=server:add(value=localhost)
```

Add the **port** configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/config-
properties=port:add(value=9000)
```

- Configure **admin-objects**.

Add an admin object.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName:add(class-
name=com.acme.eis.ra.EISAdminObjectImpl, jndi-name=java:/eis/AcmeAdminObject)
```

Configure an admin object configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName/config-
properties=threshold:add(value=10)
```

- Configure **connection-definitions**.

Add a connection definition for a managed connection factory.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-
definitions=cfName:add(class-name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-
name=java:/eis/AcmeConnectionFactory)
```

Configure a managed connection factory configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-
definitions=cfName/config-properties=name:add(value=Acme Inc)
```

Configure whether to record the enlistment trace. You can enable the recording of enlistment traces by setting the **enlistment-trace** attribute to **true**.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-
definitions=cfName:write-attribute(name=enlistment-trace,value=true)
```



WARNING

Enabling enlistment tracing makes tracking down errors during transaction enlistment easier, but comes with a performance impact.

See [Resource Adapter Attributes](#) for all available configuration options for resource adapters.

Activate the Resource Adapter

Activate the resource adapter.

```
/subsystem=resource-adapters/resource-adapter=eis.rar:activate
```

**NOTE**

You can also define capacity policies for resource adapters. For more details, see the [Capacity Policies](#) section.

16.4.3. Configure Resource Adapters to Use the Elytron Subsystem

Two types of communications occur between the server and the resource adapter in IronJacamar.

One of them is when the server opens a resource adapter connection. As defined by the specifications, this can be secured by container-managed sign-on, which requires propagation of a JAAS subject with principal and credentials to the resource adapter when opening the connection. This sign-on can be delegated to Elytron.

IronJacamar supports security inflow. This mechanism enables a resource adapter to establish security information when submitting a work to the work manager, and when delivering messages to endpoints residing in the same JBoss EAP instance.

Container-managed Sign-On

In order to achieve container-managed sign-on with Elytron, the **elytron-enabled** attribute needs to be set to **true**. This will result in all connections to the resource adapter to be secured by Elytron.

```
/subsystem=resource-adapters/resource-adapter=RAR_NAME/connection-
definitions=FACTORY_NAME:write-attribute(name=elytron-enabled,value=true)
```

The **elytron-enabled** attribute can be configured using the management CLI by setting the **elytron-enabled** attribute, in the **resource-adapters** subsystems, to **true**. By default this attribute is set to **false**.

The **authentication-context** attribute defines the name of the Elytron authentication context that will be used for performing sign-on.

An Elytron **authentication-context** attribute can contain one or more **authentication-configuration** elements, which in turn contains the credentials you want to use.

If the **authentication-context** attribute is not set, JBoss EAP will use the current **authentication-context**, which is the **authentication-context** used by the caller code that is opening the connection.

Example: Creating an authentication-configuration

```
/subsystem=elytron/authentication-configuration=exampleAuthConfig:add(authentication-
name=sa,credential-reference={clear-text=sa})
```

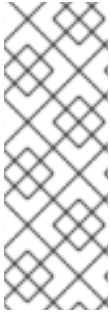
Example: Creating the authentication-context Using the above Configuration

```
/subsystem=elytron/authentication-context=exampleAuthContext:add(match-rules=[{authentication-
configuration=exampleAuthConfig}])
```

Security Inflow

It is also possible for the resource manager to inflow security credentials when submitting the work which is to be executed by the work manager. Security inflow allows the work to authenticate itself before execution. If authentication succeeds, the submitted work will be executed under the resulting authentication context. If it fails, the work execution will be rejected.

To enable Elytron security inflow, set the **wm-elytron-security-domain** attribute when configuring the resource adapter work manager. Elytron will perform the authentication based on the specified domain.



NOTE

When the resource adapter work manager is configured to use the Elytron security domain, **wm-elytron-security-domain**, the referenced work manager should have the **elytron-enabled** attribute set to **true**.

```
/subsystem=jca/workmanager=customWM:add(name=customWM, elytron-enabled=true)
```



NOTE

If instead of **wm-elytron-security-domain** the **wm-security-domain** attribute is used, the security inflow will be performed by the legacy **security** subsystem.

In the example configuration of **jca** subsystem below, we can see the configuration of a resource adapter called **ra-with-elytron-security-domain**. This resource adapter configures work manager security to use the Elytron security domain's **wm-realm**.

```
<subsystem xmlns="urn:jboss:domain:jca:5.0">
  <archive-validation enabled="true" fail-on-error="true" fail-on-warn="false"/>
  <bean-validation enabled="true"/>
  <default-workmanager>
    <short-running-threads>
      <core-threads count="50"/>
      <queue-length count="50"/>
      <max-threads count="50"/>
      <keepalive-time time="10" unit="seconds"/>
    </short-running-threads>
    <long-running-threads>
      <core-threads count="50"/>
      <queue-length count="50"/>
      <max-threads count="50"/>
      <keepalive-time time="10" unit="seconds"/>
    </long-running-threads>
  </default-workmanager>
  <workmanager name="customWM">
    <elytron-enabled>true</elytron-enabled>
    <short-running-threads>
      <core-threads count="20"/>
      <queue-length count="20"/>
      <max-threads count="20"/>
    </short-running-threads>
  </workmanager>
  <bootstrap-contexts>
    <bootstrap-context name="customContext" workmanager="customWM"/>
  </bootstrap-contexts>
  <cached-connection-manager/>
</subsystem>
```

The work manager is then referenced using the bootstrap context from the **resource-adapter** subsystem.

```

<subsystem xmlns="urn:jboss:domain:resource-adapters:5.0">
  <resource-adapters>
    <resource-adapter id="ra-with-elytron-security-domain">
      <archive>
        ra-with-elytron-security-domain.rar
      </archive>
      <bootstrap-context>customContext</bootstrap-context>
      <transaction-support>NoTransaction</transaction-support>
      <workmanager>
        <security>
          <elytron-security-domain>wm-realm</elytron-security-domain>
          <default-principal>wm-default-principal</default-principal>
          <default-groups>
            <group>
              wm-default-group
            </group>
          </default-groups>
        </security>
      </workmanager>
    </resource-adapter>
  </resource-adapters>
</subsystem>

```

Example: Configuration of the Security Domain

```

/subsystem=elytron/properties-realm=wm-properties-realm:add(users-properties={path=/security-dir/users.properties, plain-text=true}, groups-properties={path=/security-dir/groups.properties})

```

```

/subsystem=elytron/simple-role-decoder=wm-role-decoder:add(attribute=groups)

```

```

/subsystem=elytron/constant-permission-mapper=wm-permission-mapper:add(permissions=[{class-name="org.wildfly.security.auth.permission.LoginPermission"}])

```

```

/subsystem=elytron/security-domain=wm-realm:add(default-realm=wm-properties-realm, permission-mapper=wm-permission-mapper, realms=[{role-decoder=wm-role-decoder, realm=wm-properties-realm}])

```

The **Work** class is responsible for providing the credentials for Elytron's authentication under the specified domain. For that, it must implement **javax.resource.spi.work.WorkContextProvider**.

```

public interface WorkContextProvider {
    /**
     * Gets an instance of WorkContexts that needs to be used
     * by the WorkManager to set up the execution context while
     * executing a Work instance.
     *
     * @return an List of WorkContext instances.
     */
    List<WorkContext> getWorkContexts();
}

```

This interface allows the **Work** class to use the **WorkContext** to configure some aspects of the context in which the work will be executed. One of those aspects is the security inflow. For that, the **List<WorkContext> getWorkContexts** method must provide a

javax.resource.spi.work.SecurityContext. This context will use **javax.security.auth.callback.Callback** objects as defined in the [JSR-196 specification](#) to provide the credentials. The Jakarta equivalent for the authentication of the service provider interface for containers is [Jakarta Authentication](#).

Example: Creation of Callbacks Using Context

```
public class ExampleWork implements Work, WorkContextProvider {

    private final String username;
    private final String role;

    public MyWork(TestBean bean, String username, String role) {
        this.principals = null;
        this.roles = null;
        this.bean = bean;
        this.username = username;
        this.role = role;
    }

    public List<WorkContext> getWorkContexts() {
        List<WorkContext> l = new ArrayList<>(1);
        l.add(new MySecurityContext(username, role));
        return l;
    }

    public void run() {
        ...
    }

    public void release() {
        ...
    }

    public class ExampleSecurityContext extends SecurityContext {

        public void setupSecurityContext(CallbackHandler handler, Subject executionSubject, Subject
serviceSubject) {
            try {
                List<javax.security.auth.callback.Callback> cbs = new ArrayList<>();
                cbs.add(new CallerPrincipalCallback(executionSubject, new SimplePrincipal(username)));
                cbs.add(new GroupPrincipalCallback(executionSubject, new String[]{role}));
                handler.handle(cbs.toArray(new javax.security.auth.callback.Callback[cbs.size()]));
            } catch (Throwable t) {
                throw new RuntimeException(t);
            }
        }
    }
}
```

In the above example, **ExampleWork** implements the **WorkContextProvider** interface to provide **ExampleSecurityContext**. That context will create the callbacks necessary to provide the security information that will be authenticated by Elytron upon work execution.

16.4.4. Deploy and Configure the IBM MQ Resource Adapter

You can find the instructions for [deploying the IBM MQ resource adapter](#) in *Configuring Messaging* for JBoss EAP.

16.4.5. Deploy and Configure the Generic JMS Resource Adapter

You can find the instructions for [configuring the generic JMS resource adapter](#) in *Configuring Messaging* for JBoss EAP.

16.5. CONFIGURE MANAGED CONNECTION POOLS

JBoss EAP provides three implementations of the **ManagedConnectionPool** interface.

org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool

This is the default connection pool in JBoss EAP 7 and provides the best out-of-the-box performance.

org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool

This was the default connection pool in previous JBoss EAP versions.

org.jboss.jca.core.connectionmanager.pool.mcp.LeakDumperManagedConnectionPool

This connection pool is used for debugging purposes only and will report any leaks upon shutdown or when the pool is flushed.

You can set the managed connection pool implementation for a datasource using the following management CLI command.

```
/subsystem=datasources/data-source=DATA_SOURCE:write-attribute(name=mcp,value=MCP_CLASS)
```

You can set the managed connection pool implementation for a resource adapter using the following management CLI command.

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:write-attribute(name=mcp,value=MCP_CLASS)
```

You can set the managed connection pool implementation for a messaging server using the following management CLI command.

```
/subsystem=messaging-activemq/server=SERVER/pooled-connection-factory=CONNECTION_FACTORY:write-attribute(name=managed-connection-pool,value=MCP_CLASS)
```

16.6. VIEW CONNECTION STATISTICS

You can read statistics for a defined connection from the **/deployment=NAME.rar** subtree. This is so that you can access statistics for any RAR, even if it not defined in a **standalone.xml** or **domain.xml** configuration.

```
/deployment=NAME.rar/subsystem=resource-adapters/statistics=statistics/connection-definitions=java:\testMe:read-resource(include-runtime=true)
```


**NOTE**

Be sure to specify the **include-runtime=true** argument, as all statistics are runtime-only information.

See [Resource Adapter Statistics](#) for information on the available statistics.

16.7. FLUSHING RESOURCE ADAPTER CONNECTIONS

You can flush resource adapter connections using the following management CLI commands.

**NOTE**

In a managed domain, you must precede these commands with **/host=HOST_NAME/server=SERVER_NAME**.

- Flush all connections in the pool.

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-all-connection-in-pool
```

- Gracefully flush all connections in the pool.

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-gracefully-connection-in-pool
```

The server will wait until connections become idle before flushing them.

- Flush all idle connections in the pool.

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-idle-connection-in-pool
```

- Flush all invalid connections in the pool.

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-invalid-connection-in-pool
```

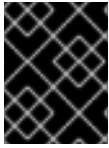
The server will flush all connections that it determines to be invalid.

16.8. TUNING THE RESOURCE ADAPTERS SUBSYSTEM

For tips on monitoring and optimizing performance for the **resource-adapters** subsystem, see the [Datasource and Resource Adapter Tuning](#) section of the *Performance Tuning Guide*.

CHAPTER 17. CONFIGURING THE WEB SERVER (UNDERTOW)

17.1. UNDERTOW SUBSYSTEM OVERVIEW



IMPORTANT

In JBoss EAP 7, the **undertow** subsystem takes the place of the **web** subsystem from JBoss EAP 6.

The **undertow** subsystem allows you to configure the web server and servlet container settings. It implements the [Jakarta Servlet 4.0 Specification](#) as well as websockets. It also supports HTTP upgrade and using high performance non-blocking handlers in servlet deployments. The **undertow** subsystem also has the ability to act as a high performance reverse proxy which supports `mod_cluster`.

Within the **undertow** subsystem, there are five main components to configure:

- [buffer caches](#)
- [server](#)
- [servlet container](#)
- [handlers](#)
- [filters](#)



NOTE

While JBoss EAP does offer the ability to update the configuration for each of these components, the default configuration is suitable for most use cases and provides reasonable performance settings.

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-
host="default-host" default-servlet-container="default" default-security-domain="other">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
    <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-
http2="true"/>
    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content"/>
      <http-invoker security-realm="ApplicationRealm"/>
    </host>
  </server>
  <servlet-container name="default">
    <jsp-config/>
    <websockets/>
  </servlet-container>
  <handlers>
```

```
<file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
</handlers>
</subsystem>
```



IMPORTANT

The **undertow** subsystem also relies on the **io** subsystem to provide XNIO workers and buffer pools. The **io** subsystem is configured separately and provides a default configuration which should give optimal performance in most cases.



NOTE

Compared to the **web** subsystem in JBoss EAP 6, the **undertow** subsystem in JBoss EAP 7 has different [default behaviors of HTTP methods](#).

Using Elytron with Undertow Subsystem

As a web application is deployed, the name of the security domain required by that application will be identified. This will either be from within the deployment or if the deployment does not have a security domain, the **default-security-domain** as defined on the **undertow** subsystem will be assumed. By default it is assumed that the security domain maps to a **PicketBox** defined in the legacy security subsystem. However, an **application-security-domain** resource can be added to the **undertow** subsystem which maps from the name of the security domain required by the application to the appropriate Elytron configuration.

Example: Adding a Mapping.

```
/subsystem=undertow/application-security-domain=ApplicationDomain:add(security-
domain=ApplicationDomain)
```

The addition of mapping is successful if the result is:

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" ... default-security-domain="other">
...
  <application-security-domains>
    <application-security-domain name="ApplicationDomain" security-domain="ApplicationDomain"/>
  </application-security-domains>
...
</subsystem>
```



NOTE

- If the deployment was already deployed at this point, the application server should be reloaded for the application security domain mapping to take effect.
- In current web service-Elytron integration, the name of the security domain specified to secure a web service endpoint and the Elytron security domain name must be the same.

This simple form is suitable where a deployment is using the standard HTTP mechanism as defined within the Servlet specification like **BASIC**, **CLIENT_CERT**, **DIGEST**, **FORM**. Here, the authentication will be performed against the **ApplicationDomain** security domain. This form is also suitable where an application is not using any authentication mechanism and instead is using programmatic authentication or is trying to obtain the **SecurityDomain** associated with the deployment and use it directly.

Example: Advanced Form of the Mapping:

```
/subsystem=undertow/application-security-domain=MyAppSecurity:add(http-authentication-
factory=application-http-authentication)
```

The advanced mapping is successful if the result is:

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" ... default-security-domain="other">
...
  <application-security-domains>
    <application-security-domain name="MyAppSecurity" http-authentication-factory="application-
http-authentication"/>
  </application-security-domains>
...
</subsystem>
```

In this form of the configuration, instead of referencing a security domain, an **http-authentication-factory** is referenced. This is the factory that will be used to obtain the instances of the authentication mechanisms and is in turn associated with the security domain.

You should reference an **http-authentication-factory** attribute when using custom HTTP authentication mechanisms or where additional configuration must be defined for mechanisms such as principal transformers, credential factories, and mechanism realms. It is also better to reference an **http-authentication-factory** attribute when using mechanisms other than the four described in the **Servlet** specification.

When the advanced form of mapping is used, another configuration option is available, **override-deployment-config**. The referenced **http-authentication-factory** can return a complete set of authentication mechanisms. By default, these are filtered to just match the mechanisms requested by the application. If this option is set to **true**, then the mechanisms offered by the factory will override the mechanisms requested by the application.

The **application-security-domain** resource also has one additional option **enable-jacc**. If this is set to **true**, JACC will be enabled for any deployments matching this mapping.

Runtime Information

Where an **application-security-domain** mapping is in use, it can be useful to double check that deployments did match against it as expected. If the resource is read with **include-runtime=true**, the deployments that are associated with the mapping will also be shown as:

```
/subsystem=undertow/application-security-domain=MyAppSecurity:read-resource(include-
runtime=true)
{
  "outcome" => "success",
  "result" => {
    "enable-jacc" => false,
    "http-authentication-factory" => undefined,
    "override-deployment-config" => false,
    "referencing-deployments" => ["simple-webapp.war"],
    "security-domain" => "ApplicationDomain",
    "setting" => undefined
  }
}
```

In this output, the **referencing-deployments** attribute shows that the deployment **simple-webapp.war** has been deployed using the mapping.

17.2. CONFIGURING BUFFER CACHES

The buffer cache is used to cache static resources. JBoss EAP enables multiple caches to be configured and referenced by deployments, allowing different deployments to use different cache sizes. Buffers are allocated in regions and are a fixed size. The total amount of space used can be calculated by multiplying the buffer size by the number of buffers per region by the maximum number of regions. The default size of a buffer cache is 10MB.

JBoss EAP provides a single cache by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-
host="default-host" default-servlet-container="default" default-security-domain="other">
  <buffer-cache name="default"/>
  ...
</subsystem>
```

Updating an Existing Buffer Cache

To update an existing buffer cache:

```
/subsystem=undertow/buffer-cache=default/:write-attribute(name=buffer-size,value=2048)
```

```
reload
```

Creating a New Buffer Cache

To create a new buffer cache:

```
/subsystem=undertow/buffer-cache=new-buffer:add
```

Deleting a Buffer Cache

To delete a buffer cache:

```
/subsystem=undertow/buffer-cache=new-buffer:remove
```

```
reload
```

For a full list of the attributes available for configuring buffer caches, please see the [Undertow Subsystem Attributes](#) section.

17.3. CONFIGURING BYTE BUFFER POOLS

Undertow byte buffer pools are used to allocate pooled NIO **ByteBuffer** instances. All listeners have a byte buffer pool and you can use different buffer pools and workers for each listener. Byte buffer pools can be shared between different server instances.

These buffers are used for IO operations, and the buffer size has a big impact on application performance. For most servers, the ideal size is usually 16k.

Updating an Existing Byte Buffer Pool

To update an existing byte buffer pool:

```
/subsystem=undertow/byte-buffer-pool=myByteBufferPool:write-attribute(name=buffer-size,value=1024)
```

```
reload
```

Creating a Byte Buffer Pool

To create a new byte buffer pool:

```
/subsystem=undertow/byte-buffer-pool=newByteBufferPool:add
```

Deleting a Byte Buffer Pool

To delete a byte buffer pool:

```
/subsystem=undertow/byte-buffer-pool=newByteBufferPool:remove
```

```
reload
```

For a full list of the attributes available for configuring byte buffer pools, see the [Byte Buffer Pool Attributes](#) reference.

17.4. CONFIGURING A SERVER

A server represents an instance of Undertow and consists of several elements:

- host
- http-listener
- https-listener
- ajp-listener

The host element provides a virtual host configuration, while the three listeners provide connections of that type to the Undertow instance.

The default behavior of the server is to queue requests while the server is starting. You can change the default behavior using the **queue-requests-on-start** attribute on the host. If this attribute is set to **true**, which is the default, then requests that arrive when the server is starting will be held until the server is ready. If this attribute is set to **false**, then requests that arrive before the server has completely started will be rejected with the default response code.

Regardless of the attribute value, request processing does not start until the server is completely started.

You can configure the **queue-requests-on-start** attribute using the management console by navigating to **Configuration → Subsystems → Web (Undertow) → Server**, selecting the server and clicking **View**, and selecting the **Hosts** tab. For a managed domain, you must specify which profile to configure.



NOTE

Multiple servers can be configured, allowing deployments and servers to be completely isolated. This can be useful in certain scenarios such as multi-tenant environments.

JBoss EAP provides a server by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-
host="default-host" default-servlet-container="default" default-security-domain="other">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
    <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-
http2="true"/>
    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content"/>
      <http-invoker security-realm="ApplicationRealm"/>
    </host>
  </server>
  ...
</subsystem>
```

The following examples show how to configure a server using the management CLI. You can also configure a server using the management console by navigating to **Configuration → Subsystems → Web (Undertow) → Server**.

Updating an Existing Server

To update an existing server:

```
/subsystem=undertow/server=default-server:write-attribute(name=default-host,value=default-host)
```

```
reload
```

Creating a New Server

To create a new server:

```
/subsystem=undertow/server=new-server:add
```

```
reload
```

Deleting a Server

To delete a server:

```
/subsystem=undertow/server=new-server:remove
```

```
reload
```

For a full list of the attributes available for configuring servers, see the [Undertow Subsystem Attributes](#) section.

17.4.1. Access Logging

You can configure access logging on each host you define.

Two access logging options are available: standard access logging and console access logging.

Note that the additional processing required for access logging can affect system performance.

17.4.1.1. Standard Access Logging

Standard access logging writes log entries to a log file.

By default, the log file is stored in the directory `standalone/log/access_log.log`.

To enable standard access logging, add the `access-log` setting to the host for which you want to capture access log data. The following CLI command illustrates the configuration on the default host in the default JBoss EAP server:

```
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:add
```



NOTE

You must reload the server after enabling standard access logging.

By default, the access log record includes the following data:

- Remote host name
- Remote logical user name (always -)
- Remote user that was authenticated
- The date and time of the request, in Common Log Format
- The first line of the request
- The HTTP status code of the response
- The number of bytes sent, excluding HTTP headers

This set of data is defined as the common pattern. Another pattern, `combined`, is also available. In addition to the data logged in the common pattern, the `combined` pattern includes the `referer` and `user agent` from the incoming header.

You can change the data logged using the **pattern** attribute. The following CLI command illustrates updating the **pattern** attribute to use the `combined` pattern:

```
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:write-attribute(name=pattern,value="combined")
```



NOTE

You must reload the server after updating the pattern attribute.

Table 17.1. Available patterns

Pattern	Description
%a	Remote IP address
%A	Local IP address
%b	Bytes sent, excluding HTTP headers or - if no bytes were sent
%B	Bytes sent, excluding HTTP headers
%h	Remote host name
%H	Request protocol
%l	Remote logical username from identd (always returns -; included for Apache access log compatibility)
%m	Request method
%p	Local port
%q	Query string (excluding the ? character)
%r	First line of the request
%s	HTTP status code of the response
%t	Date and time, in Common Log Format format
%u	Remote user that was authenticated
%U	Requested URL path
%v	Local server name
%D	Time taken to process the request, in milliseconds
%T	Time taken to process the request, in seconds
%I	Current Request thread name (can compare later with stack traces)
common	%h %l %u %t "%r" %s %b

Pattern	Description
combined	<code>%h %l %u %t "%r" %s %b "%{i,Referer}" "%{i,User-Agent}"</code>

You can also write information from the cookie, the incoming header and response header, or the session. The syntax is modeled after the Apache syntax:

- `%{i,xxx}` for incoming headers
- `%{o,xxx}` for outgoing response headers
- `%{c,xxx}` for a specific cookie
- `%{r,xxx}` where **xxx** is an attribute in the **ServletRequest**
- `%{s,xxx}` where **xxx** is an attribute in the **HttpSession**

Additional configuration options are available for this log. For more information see "access-log Attributes" in the appendix.

17.4.1.2. Console Access Logging

Console access logging writes data to stdout as structured as JSON data.

Each access log record is a single line of data. You can capture this data for processing by log aggregation systems.

To configure console access logging, add the console-access-log setting to the host for which you want to capture access log data. The following CLI command illustrates the configuration on the default host in the default JBoss EAP server:

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:add
```

By default, the console access log record includes the following data:

Table 17.2. Default console access log data

Log data field name	Description
eventSource	The source of the event in the request
hostName	The JBoss EAP host that processed the request
bytesSent	The number of bytes the JBoss EAP server sent in response to the request
dateTime	The date and time that the request was processed by the JBoss EAP server
remoteHost	The IP address of the machine where the request originated

Log data field name	Description
remoteUser	The user name associated with the remote request
requestLine	The request submitted
responseCode	The HTTP response code returned by the JBoss EAP server

Default properties are always included in the log output. You can use the **attributes** attribute to change the labels of the default log data, and in some cases to change the data configuration. You can also use the **attributes** attribute to add additional log data to the output.

Table 17.3. Available console access log data

Log data field name	Description	Format
authentication-type	The authentication type used to authenticate the user associated with the request. Default label: authenticationType Use the key option to change the label for this property.	authentication-type{} authentication-type={key="authType"}
bytes-sent	The number of bytes returned for the request, excluding HTTP headers. Default label: bytesSent Use the key option to change the label for this property.	bytes-sent={} bytes-sent={key="sent-bytes"}
date-time	The date and time that the request was received and processed. Default label: dateTime Use the key option to change the label for this property. Use the date-format to define the pattern used to format the date-time record. The pattern must be a Java SimpleDateFormat pattern. Use the time-zone option to specify the time zone used to format the date and/or time data if the date-format option is defined. This value must be a valid java.util.TimeZone.	date-time={key="<keyname>", date-format="<date-time format>" } date-time= {key="@timestamp", date- format="yyyy-MM- dd'T'HH:mm:ssSSS"}

Log data field name	Description	Format
host-and-port	The host and port queried by the request. Default label: hostAndPort Use the key option to change the label for this property.	host-and-port{} host-and-port={key="port-host"}
local-ip	The IP address of the local connection. Use the key option to change the label for this property. Default label: localIp Use the key option to change the label for this property.	local-ip{} local-ip{key="localIP"}
local-port	The port of the local connection. Default label: localPort Use the key option to change the label for this property.	local-port{} local-port{key="LocalPort"}
local-server-name	The name of the local server that processed the request. Default label: localServerName Use the key option to change the label for this property.	local-server-name {} local-server-name {key=LocalServerName}
path-parameter	One or more path or URI parameters included in the request. The names property is a comma-separated list of names used to resolve the exchange values. Use the key-prefix property to make the keys unique. If the key-prefix is specified, the prefix is prepended to the name of each path parameter in the output.	path-parameter{names={store,section}} path-parameter{names={store,section}, key-prefix="my-"}
predicate	The name of the predicate context. The names property is a comma-separated list of names used to resolve the exchange values. Use the key-prefix property to make the keys unique. If the key-prefix is specified, the prefix is prepended to the name of each path parameter in the output.	predicate{names={store,section}} predicate{names={store,section}, key-prefix="my-"}

Log data field name	Description	Format
query-parameter	One or query parameters included in the request. The names property is a comma-separated list of names used to resolve the exchange values. Use the key-prefix property to make the keys unique. If the key-prefix is specified, the prefix is prepended to the name of each path parameter in the output.	query-parameter{names={store,section}} query-parameter{names={store,section}, key-prefix="my-"}
query-string	The query string of the request. Default label: queryString Use the key option to change the label for this property. Use the include-question-mark property to specify whether the query string should include the question mark. By default, the question mark is not included.	query-string{} query-string{key="QueryString", include-question-mark="true"}
relative-path	The relative path of the request. Default label: relativePath Use the key option to change the label for this property.	relative-path{} relative-path{key="RelativePath"}
remote-host	The remote host name. Default label: remoteHost Use the key option to change the label for this property.	remote-host{} remote-host{key="RemoteHost"}
remote-ip	The remote IP address. Default label: remoteIp Use the key options to change the label for this property. Use the obfuscated property to obfuscate the IP address in the output log record. The default value is false.	remote-ip{} remote-ip{key="RemoteIP", obfuscated="true"}
remote-user	Remote user that was authenticated. Default label: remoteUser Use the key options to change the label for this property.	remote-user{} remote-user{key="RemoteUser"}

Log data field name	Description	Format
request-header	The name of a request header. The key for the structured data is the name of the header; the value is the value of the named header. The names property is a comma-separated list of names used to resolve the exchange values. Use the key-prefix property to make the keys unique. If the key-prefix is specified, the prefix is prepended to the name of the request headers in the log output.	request-header{names={store,section}} request-header{names={store,section}, key-prefix="my-"}
request-line	The request line. Default label: requestLine Use the key option to change the label for this property.	request-line{} request-line{key="Request-Line"}
request-method	The request method. Default label: requestMethod Use the key option to change the label for this property.	request-method{} request-method{key="RequestMethod"}
request-path	The relative path for the request. Default label: requestPath Use the key option to change the label for this property.	request-path{} request-path{key="RequestPath"}
request-protocol	The protocol for the request. Default label: requestProtocol Use the key option to change the label for this property.	request-protocol{} request-protocol{key="RequestProtocol"}
request-scheme	The URI scheme of the request. Default label: requestScheme Use the key option to change the label for this property.	request-scheme{} request-scheme{key="RequestScheme"}
request-url	The original request URI. Includes host name, protocol, and so forth, if specified by the client. Default label: requestUrl Use the key option to change the label for this property.	request-url{} request-url{key="RequestURL"}
resolved-path	The resolved path. Default Label: resolvedPath Use the key option to change the label for this property.	resolved-path{} resolved-path{key="ResolvedPath"}

Log data field name	Description	Format
response-code	The response code. Default label: <code>responseCode</code> Use the key option to change the label for this property.	<code>response-code{} response-code{key="ResponseCode"}</code>
response-header	The name of a response header. The key for the structured data is the name of the header; the value is the value of the named header. The names property is a comma-separated list of names used to resolve the exchange values. Use the key-prefix property to make the keys unique. If the key-prefix is specified, the prefix is prepended to the name of the request headers in the log output.	<code>response-header{names={store,section}} response-header{names={store,section}, key-prefix="my-"}</code>
response-reason-phrase	The text reason for the response code. Default label: <code>responseReasonPhrase</code> Use the key option to change the label for this property.	<code>response-reason-phrase{} response-reason-phrase{key="ResponseReasonPhrase"}</code>
response-time	The time used to process the request. Default label: <code>responseTime</code> Use the key option to change the label for this property. The default time unit is <code>MILLISECONDS</code> . Available time units include: <code>* NANOSECONDS *</code> <code>* MICROSECONDS *</code> <code>* MILLISECONDS *</code> <code>* SECONDS</code>	<code>response-time{} response-time{key="ResponseTime", time-unit=SECONDS}</code>
secure-exchange	Indicates whether the exchange was secure. Default label: <code>secureExchange</code> Use the key option to change the label for this property.	<code>secure-exchange{} secure-exchange{key="SecureExchange"}</code>
ssl-cipher	The SSL cipher for the request. Default label: <code>sslCipher</code> Use the key option to change the label for this property.	<code>ssl-cipher{} ssl-cipher{key="SSLCipher"}</code>
ssl-client-cert	The SSL client certificate for the request. Default label: <code>sslClientCert</code> Use the key option to change the label for this property.	<code>ssl-client-cert{} ssl-client-cert{key="SSLClientCert"}</code>

Log data field name	Description	Format
ssl-session-id	The SSL session id of the request. Default label: sslSessionId Use the key option to change the label for this property.	ssl-session-id{} stored-response
The stored response to the request. Default label: storedResponse Use the key option to change the label for this property.	stored-response{} stored-response{key="StoredResponse"}	thread-name
The thread name of the current thread. Default label: threadName Use the key option to change the label for this property.	thread-name{} thread-name{key="ThreadName"}	transport-protocol

You can use the **metadata** attribute to configure additional arbitrary data to include in the access log record. The value of the **metadata** attribute is a set of key:value pairs that defines the data to include in the access log record. The value in a pair can be a management model expression. Management model expressions are resolved when the server is started or reloaded. Key-value pairs are comma-separated.

The following CLI command demonstrates an example of a complex console log configuration, including additional log data, customization of log data, and additional metadata:

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:add(metadata={"@version"="1", "qualifiedHostName"=${jboss.qualified.host.name:unknown}}, attributes={bytes-sent={}, date-time={key="@timestamp", date-format="yyyy-MM-dd'T'HH:mm:ssSSS"}, remote-host={}, request-line={}, response-header={key-prefix="responseHeader", names=["Content-Type"]}, response-code={}, remote-user={}})
```

The resulting access log record would resemble the following additional JSON data (Note: the example output below is formatted for readability; in an actual record, all data would be output as a single line):

```
{
  "eventSource":"web-access",
  "hostName":"default-host",
  "@version":"1",
  "qualifiedHostName":"localhost.localdomain",
  "bytesSent":1504,
  "@timestamp":"2019-05-02T11:57:37123",
  "remoteHost":"127.0.0.1",
  "remoteUser":null,
  "requestLine":"GET / HTTP/2.0",
  "responseCode":200,
  "responseHeaderContent-Type":"text/html"
}
```

The following command illustrates updates to the log data after activating the console access log:

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:write-
```



```
attribute(name=attributes,value={bytes-sent={}, date-time={key="@timestamp", date-format="yyyy-MM-dd'T'HH:mm:ssSSS"}, remote-host={}, request-line={}, response-header={key-prefix="responseHeader", names=["Content-Type"]}, response-code={}, remote-user={}})
```

The following command illustrates updates to the custom metadata after activating the console access log:

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:write-attribute(name=metadata,value={"@version"="1", "qualifiedHostName"=${jboss.qualified.host.name:unknown}})
```

17.5. CONFIGURING A SERVLET CONTAINER

A servlet container provides all servlet, JSP and websocket-related configuration, including session-related configuration. While most servers will only need a single servlet container, it is possible to configure multiple servlet containers by adding an additional **servlet-container** element. Having multiple servlet containers enables behavior such as allowing multiple deployments to be deployed to the same context path on different virtual hosts.



NOTE

Much of the configuration provided in by servlet container can be individually overridden by deployed applications using their **web.xml** file.

JBoss EAP provides a servlet container by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
  <buffer-cache name="default"/>
  <server name="default-server">
    ...
  </server>
  <servlet-container name="default">
    <jsp-config/>
    <websockets/>
  </servlet-container>
  ...
</subsystem>
```

The following examples show how to configure a servlet container using the management CLI. You can also configure a servlet container using the management console by navigating to **Configuration → Subsystems → Web (Undertow) → Servlet Container**.

Updating an Existing Servlet Container

To update an existing servlet container:

```
/subsystem=undertow/servlet-container=default:write-attribute(name=ignore-flush,value=true)
```

```
reload
```

Creating a New Servlet Container

To create a new servlet container:

```
/subsystem=undertow/servlet-container=new-servlet-container:add
```

```
reload
```

Deleting a Servlet Container

To delete a servlet container:

```
/subsystem=undertow/servlet-container=new-servlet-container:remove
```

```
reload
```

For a full list of the attributes available for configuring servlet containers, see the [Undertow Subsystem Attributes](#) section.

17.6. CONFIGURING A SERVLET EXTENSION

Servlet extensions allow you to hook into the servlet deployment process and modify aspects of a servlet deployment. This can be useful in cases where you need to add additional authentication mechanisms to a deployment or use native Undertow handlers as part of a servlet deployment.

To create a custom servlet extension, it is necessary to implement the **io.undertow.servlet.ServletExtension** interface and then add the name of your implementation class to the **META-INF/services/io.undertow.servlet.ServletExtension** file in the deployment. You also need to include the compiled class file of the **ServletExtension** implementation. When Undertow deploys the servlet, it loads all the services from the **deployments** class loader and then invokes their **handleDeployment** methods.

An Undertow **DeploymentInfo** structure, which contains a complete and mutable description of the deployment, is passed to this method. You can modify this structure to change any aspect of the deployment.

The **DeploymentInfo** structure is the same structure that is used by the embedded API, so in effect a **ServletExtension** has the same amount of flexibility that you have when using Undertow in embedded mode.

17.7. CONFIGURING HANDLERS

JBoss EAP allows for two types of handlers to be configured:

- file handlers
- reverse-proxy handlers

File handlers serve static files. Each file handler must be attached to a location in a virtual host. Reverse-proxy handlers allow JBoss EAP to serve as a high performance reverse-proxy.

JBoss EAP provides a file handler by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-
host="default-host" default-servlet-container="default" default-security-domain="other">
  <buffer-cache name="default"/>
```

```

<server name="default-server">
  ...
</server>
<servlet-container name="default">
  ...
</servlet-container>
<handlers>
  <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
</handlers>
</subsystem>

```

Using WebDAV for Static Resources

Previous versions of JBoss EAP allowed for using WebDAV with the **web** subsystem, by way of the **WebdavServlet**, to host static resources and enable additional HTTP methods for accessing and manipulating those files. In JBoss EAP 7, the **undertow** subsystem does provide a mechanism for serving static files using a file handler, but the **undertow** subsystem does not support WebDAV. If you want to use WebDAV with JBoss EAP 7, you can write a custom WebDAV servlet.

Updating an Existing File Handler

To update an existing file handler:

```
/subsystem=undertow/configuration=handler/file=welcome-content:write-attribute(name=case-sensitive,value=true)
```

```
reload
```

Creating a New File Handler

To create a new file handler:

```
/subsystem=undertow/configuration=handler/file=new-file-handler:add(path="${jboss.home.dir}/welcome-content")
```



WARNING

If you set a file handler's **path** directly to a file instead of a directory, any **location** elements that reference that file handler must not end with a forward slash (/). Otherwise, the server will return a **404 - Not Found** response.

Deleting a File Handler

To delete a file handler

```
/subsystem=undertow/configuration=handler/file=new-file-handler:remove
```

```
reload
```

For a full list of the attributes available for configuring handlers, see the [Undertow Subsystem Attributes](#) section.

17.8. CONFIGURING FILTERS

A filter enables some aspect of a request to be modified and can use predicates to control when a filter executes. Some common use cases for filters include setting headers or doing GZIP compression.



NOTE

A filter is functionally equivalent to a global valve used in JBoss EAP 6.

The following types of filters can be defined:

- custom-filter
- error-page
- expression-filter
- gzip
- mod-cluster
- request-limit
- response-header
- rewrite

The following examples show how to configure a filter using the management CLI. You can also configure a filter using the management console by navigating to **Configuration → Subsystems → Web (Undertow) → Filters**.

Updating an Existing Filter

To update an existing filter:

```
/subsystem=undertow/configuration=filter/response-header=myHeader:write-attribute(name=header-value,value="JBoss-EAP")
```

```
reload
```

Creating a New Filter

To create a new filter:

```
/subsystem=undertow/configuration=filter/response-header=new-response-header:add(header-name=new-response-header,header-value="My Value")
```

Deleting a Filter

To delete a filter:

```
/subsystem=undertow/configuration=filter/response-header=new-response-header:remove
```

```
reload
```

For a full list of the attributes available for configuring filters, see the [Undertow Subsystem Attributes](#) section.

17.8.1. Configuring the buffer-request Handler

A request from the client or the browser consists of two parts: the header and the body. In a typical situation, the header and the body are sent to JBoss EAP without any delays in between. However, if the header is sent first and then after few seconds, the body is sent, there is a delay sending the complete request. This scenario creates a thread in JBoss EAP to show as **waiting** to execute the complete request.

The delay caused in sending the header and the body of the request can be corrected using the **buffer-request** handler. The **buffer-request** handler attempts to consume the request from a non-blocking IO thread before allocating it to a worker thread. When no **buffer-request** handler is added, the thread allocation to the worker thread happens directly. However, when the **buffer-request** handler is added, the handler attempts to read the amount of data that it can buffer in a non-blocking manner using the IO thread before allocating it to the worker thread.

You can use the following management CLI commands to configure the **buffer-request** handler:

```
/subsystem=undertow/configuration=filter/expression-filter=buf:add(expression="buffer-request(buffers=1)")
```

```
/subsystem=undertow/server=default-server/host=default-host/filter-ref=buf:add
```

There is a limit to the size of the buffer requests that can be processed. This limit is determined by a combination of the buffer size and the total number of buffers, as shown in the equation below.

$$\text{Total_size} = \text{num_buffers} \times \text{buffer_size}$$

In the equation above:

- **Total_size** is the size of data that will be buffered before the request is dispatched to a worker thread.
- **num_buffers** is the number of buffers. The number of buffers is set by the **buffers** parameter on the handler.
- **buffer_size** is the size of each buffer. The buffer size is set in the **io** subsystem, and is 16KB by default per request.



WARNING

Avoid configuring very large buffer requests, or else you might run out of memory.

17.9. CONFIGURE THE DEFAULT WELCOME WEB APPLICATION

JBoss EAP includes a default **Welcome** application, which displays at the root context on port **8080** by default.

There is a default server preconfigured in Undertow that serves up the welcome content.

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-
host="default-host" default-servlet-container="default" default-security-domain="other">
  ...
  <server name="default-server">
    <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
    <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-
http2="true"/>
    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content"/>
      <http-invoker security-realm="ApplicationRealm"/>
    </host>
  </server>
  ...
  <handlers>
    <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
  </handlers>
</subsystem>
```

The default server, **default-server**, has a default host, **default-host**, configured. The default host is configured to handle requests to the server's root, using the **<location>** element, with the **welcome-content** file handler. The **welcome-content** handler serves up the content in the location specified in the **path** property.

This default **Welcome** application can be replaced with your own web application. This can be configured in one of two ways:

- [Change the **welcome-content** file handler](#)
- [Change the **default-web-module**](#)

You can also [disable the welcome content](#).

Change the welcome-content File Handler

1. Modify the existing **welcome-content** file handler's path to point to the new deployment.

```
/subsystem=undertow/configuration=handler/file=welcome-content:write-
attribute(name=path,value="/path/to/content")
```



NOTE

Alternatively, you could create a different file handler to be used by the server's root.

```
/subsystem=undertow/configuration=handler/file=NEW_FILE_HANDLER:add(
path="/path/to/content")
/subsystem=undertow/server=default-server/host=default-
host/location=/:write-attribute(name=handler,value=NEW_FILE_HANDLER)
```

2. Reload the server for the changes to take effect.

```
reload
```

Change the default-web-module

1. Map a deployed web application to the server's root.

```
/subsystem=undertow/server=default-server/host=default-host:write-attribute(name=default-web-module,value=hello.war)
```

2. Reload the server for the changes to take effect.

```
reload
```

Disable the Default Welcome Web Application

1. Disable the welcome application by removing the **location** entry / for the **default-host**.

```
/subsystem=undertow/server=default-server/host=default-host/location=/:remove
```

2. Reload the server for the changes to take effect.

```
reload
```

17.10. CONFIGURING HTTPS

For information on configuring HTTPS for web applications, see [Configure One-way and Two-way SSL/TLS for Applications](#) in *How to Configure Server Security*.

For information on configuring HTTPS for use with the JBoss EAP management interfaces, see [How to Secure the Management Interfaces](#) in *How to Configure Server Security*.

17.11. CONFIGURING HTTP SESSION TIMEOUT

The HTTP session timeout defines the period of inactive time needed to declare an HTTP session invalid. For example, a user accesses an application deployed to JBoss EAP which creates an HTTP session. If that user then attempts to access that application again after the HTTP session timeout, the original HTTP session will be invalidated and the user will be forced to create a new HTTP session. This may result in the loss of unpersisted data or the user having to reauthenticate.

The HTTP session timeout is configured in an application's **web.xml** file, but a default HTTP session timeout can be specified within JBoss EAP. The server's timeout value will apply to all deployed applications, but an application's **web.xml** will override the server's value.

The server value is specified in the **default-session-timeout** property which is found in the **servlet-container** section of the **undertow** subsystem. The value of **default-session-timeout** is specified in minutes and the default is **30**.

Configuring the Default Session Timeout

To configure the **default-session-timeout**:

```
/subsystem=undertow/servlet-container=default:write-attribute(name=default-session-timeout,value=60)
```

```
reload
```

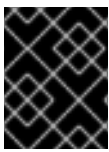
17.12. CONFIGURING HTTP-ONLY SESSION MANAGEMENT COOKIES

Session management cookies can be accessed by both HTTP APIs and non-HTTP APIs such as JavaScript. JBoss EAP offers the ability to send the **HttpOnly** header as part of the **Set-Cookie** response header to the client, usually a browser. In supported browsers, enabling this header tells the browser that it should prevent accessing session management cookies through non-HTTP APIs. Restricting session management cookies to only HTTP APIs can help to mitigate the threat of session cookie theft via cross-site scripting attacks. To enable this behavior, the **http-only** attribute should be set to **true**.



IMPORTANT

Using the **HttpOnly** header does not actually prevent cross-site scripting attacks by itself, it merely notifies the browser. The browser must also support **HttpOnly** for this behavior to take effect.



IMPORTANT

Using the **http-only** attribute only applies the restriction to session management cookies and not other browser cookies.

The **http-only** attribute is set in two places in the **undertow** subsystem:

- In the servlet container as a session cookie setting
- In the host section of the server as a single sign-on property

Configuring **host-only** for the Servlet Container Session Cookie

To configure the **host-only** property for the servlet container session cookie:

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:add
```

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:write-attribute(name=http-only,value=true)
```

```
reload
```

Configuring **host-only** for the Host Single Sign-On

To configure the **host-only** property for the host single sign-on:

```
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:add
```

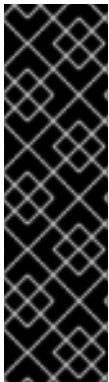
```
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:write-attribute(name=http-only,value=true)
```

```
reload
```

17.13. CONFIGURING HTTP/2

Undertow allows for the use of the [HTTP/2](#) standard, which reduces latency by compressing headers and multiplexing many streams over the same TCP connection. It also provides the ability for a server to push resources to the client before it has requested them, leading to faster page loads.

Be aware that HTTP/2 only works with clients and browsers that also support the HTTP/2 standard.



IMPORTANT

Most modern browsers enforce HTTP/2 over a secured TLS connection, known as **h2** and may not support HTTP/2 over plain HTTP, known as **h2c**. It is still possible to configure JBoss EAP to use HTTP/2 with **h2c**, in other words, without using HTTPS and only using plain HTTP with HTTP upgrade. In that case, you can simply enable HTTP/2 in the HTTP listener Undertow:

```
/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=enable-http2,value=true)
```

To configure Undertow to use HTTP/2, enable the HTTPS listener in Undertow to use HTTP/2 by setting the **enable-http2** attribute to **true**:

```
/subsystem=undertow/server=default-server/https-listener=https:write-attribute(name=enable-http2,value=true)
```

For more information on the HTTPS listener and configuring Undertow to use HTTPS for web applications, see [Configure One-way and Two-way SSL/TLS for Applications](#) in *How to Configure Server Security*.



NOTE

In order to utilize HTTP/2 with the **elytron** subsystem, you will need to ensure that the configured **ssl-context** in the **https-listener** of the Undertow is configured as modifiable. This can be achieved by setting the **wrap** attribute of the appropriate **server-ssl-context** to **false**. By default, the **wrap** attribute is set to **false**. This is required by Undertow to make modifications in the **ssl-context** about the ALPN. If the provided **ssl-context** is not writable, ALPN cannot be used and the connection falls back to HTTP/1.1.

ALPN support when using HTTP/2

When using HTTP/2 over a secured TLS connection, a TLS stack that supports ALPN TLS protocol extension is required. Obtaining this stack varies based on the installed JDK.

- When using Java 8, the ALPN implementation is introduced directly into JBoss EAP with dependencies on Java internals. Therefore, this ALPN implementation only works with Oracle and OpenJDK. It does not work with IBM Java. Red Hat strongly recommends that you utilize ALPN TLS protocol extension support from the OpenSSL provider in JBoss EAP, with OpenSSL libraries that implement ALPN capability. Using the ALPN TLS protocol extension support from the OpenSSL provider should result in better performance.
- As of Java 9, the JDK supports ALPN natively; however, using the ALPN TLS protocol extension support from the OpenSSL provider should also result in better performance when using Java 9 or later.

Instructions for installing OpenSSL, to obtain the ALPN TLS protocol extension support, are available in [Install OpenSSL from JBoss Core Services](#). The standard system OpenSSL is supported on Red Hat Enterprise Linux 8 and no additional JBoss Core Services OpenSSL is required.

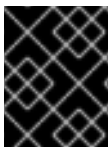
Once OpenSSL has been installed, follow the instructions in [Configure JBoss EAP to Use OpenSSL](#).

Verify HTTP/2 is Being Used

To verify that Undertow is using HTTP/2, you will need to inspect the headers coming from Undertow. Navigate to your JBoss EAP instance using https, for example <https://localhost:8443>, and use your browser's developer tools to inspect the headers. Some browsers, for example Google Chrome, will show HTTP/2 pseudo headers, such as **:path**, **:authority**, **:method** and **:scheme**, when using HTTP/2. Other browsers, for example Firefox and Safari, will report the status or version of the header as **HTTP/2.0**.

17.14. CONFIGURING A REQUESTDUMPING HANDLER

The **RequestDumping** handler, **io.undertow.server.handlers.RequestDumpingHandler**, logs the details of a request and corresponding response objects handled by Undertow within JBoss EAP.



IMPORTANT

While this handler can be useful for debugging, it may also log sensitive information. Please keep this in mind when enabling this handler.



NOTE

The **RequestDumping** handler replaces the **RequestDumperValve** from JBoss EAP 6.

You can configure a **RequestDumping** handler at either at the server level directly in JBoss EAP or within an individual application.

17.14.1. Configuring a RequestDumping Handler on the Server

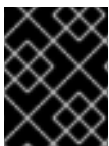
A **RequestDumping** handler should be configured as an expression filter. To configure a **RequestDumping** handler as an expression filter, you need to do the following:

Create a new Expression Filter with the **RequestDumping** Handler

```
/subsystem=undertow/configuration=filter/expression-
filter=requestDumperExpression:add(expression="dump-request")
```

Enable the Expression Filter in the Undertow Web Server

```
/subsystem=undertow/server=default-server/host=default-host/filter-
ref=requestDumperExpression:add
```



IMPORTANT

All requests and corresponding responses handled by the Undertow web server will be logged when enabling the **RequestDumping** handler as a expression filter in this manner.

Configuring a RequestDumping Handler for Specific URLs

In addition to logging all requests, you can also use an expression filter to only log requests and corresponding responses for specific URLs. This can be accomplished using a predicate in your expression such as **path**, **path-prefix**, or **path-suffix**. For example, if you want to log all requests and corresponding responses to **/myApplication/test**, you can use the expression

"path(/myApplication/test) -> dump-request" instead of the expression **"dump-request"** when creating your expression filter. This will only direct requests with a path exactly matching **/myApplication/test** to the **RequestDumping** handler.

17.14.2. Configuring a RequestDumping Handler within an Application

In addition to configuring a **RequestDumping** handler at the server, you can also configure it within individual applications. This will limit the scope of the handler to only that specific application. A **RequestDumping** handler should be configured in **WEB-INF/undertow-handlers.conf**.

To configure the **RequestDumping** handler in **WEB-INF/undertow-handlers.conf** to log all requests and corresponding responses for this application, add the following expression to **WEB-INF/undertow-handlers.conf**:

Example: WEB-INF/undertow-handlers.conf

```
dump-request
```

To configure the **RequestDumping** handler in **WEB-INF/undertow-handlers.conf** to only log requests and corresponding responses to specific URLs within this application, you can use a predicate in your expression such as **path**, **path-prefix**, or **path-suffix**. For example, to log all requests and corresponding responses to **test** in your application, the following expression with the **path** predicate could be used:

Example: WEB-INF/undertow-handlers.conf

```
path(/test) -> dump-request
```



NOTE

When using the predicates such as **path**, **path-prefix**, or **path-suffix** in expressions defined in the application's **WEB-INF/undertow-handlers.conf**, the value used will be relative to the context root of the application. For example, if the application's context root is **myApplication** with an expression **path(/test) -> dump-request** configured in **WEB-INF/undertow-handlers.conf**, it will only log requests and corresponding responses to **/myApplication/test**.

17.15. CONFIGURING COOKIE SECURITY

You can use the **secure-cookie** handler to enhance the security of cookies that are created over a connection between a server and a client. In this case, if the connection over which the cookie is set is marked as secure, the cookie will have its **secure** attribute set to **true**.

You can secure the connection by configuring a listener or by using HTTPS. You configure the **secure-cookie** handler by defining an **expression-filter** in the **undertow** subsystem. For more information, see [Configuring Filters](#).

When the **secure-cookie** handler is in use, cookies that are set over a secure connection will be implicitly set as secure and will never be sent over an unsecure connection.

17.16. TUNING THE UNDERTOW SUBSYSTEM

For tips on optimizing performance for the **undertow** subsystem, see the [Undertow Subsystem Tuning](#) section of the *Performance Tuning Guide*.

CHAPTER 18. CONFIGURING REMOTING

18.1. ABOUT THE REMOTING SUBSYSTEM

The **remoting** subsystem allows you to configure inbound and outbound connections for local and remote services as well as the settings for those connections.

JBoss Remoting includes the following configurable elements: the endpoint, connectors, and a series of local and remote connection URIs. Most people will not need to configure the **remoting** subsystem at all, unless they use custom connectors for their own applications. Applications that act as remoting clients, such as EJBs, need separate configuration to connect to a specific connector.

Default Remoting Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:remoting:4.0">
  <endpoint/>
  <http-connector name="http-remoting-connector" connector-ref="default" security-
realm="ApplicationRealm"/>
</subsystem>
```

See [Remoting Subsystem Attributes](#) for a full list of the attributes available for the **remoting** subsystem.

The Remoting Endpoint

The remoting endpoint uses the XNIO worker declared and configured by the **io** subsystem.

See [Configuring the Endpoint](#) for details on how to configure the remoting endpoint.

Connector

The connector is the main remoting configuration element. Multiple connectors are allowed. Each connector consists of a **<connector>** element with several sub-elements, and few other attributes. The default connector is used by several JBoss EAP subsystems. Specific settings for the elements and attributes of your custom connectors depend on your applications. Contact Red Hat Global Support Services for more information.

See [Configuring a Connector](#) for details on how to configure connectors.

Outbound Connections

You can specify three different types of outbound connections:

- An [outbound connection](#), specified by a URI
- A [local outbound connection](#), which connects to a local resource such as a socket
- A [remote outbound connection](#), which connects to a remote resource and authenticates using a security realm

Additional Configuration

Remoting depends on several elements that are configured outside of the **remoting** subsystem, such as the network interface and IO worker.

For more information, see [Additional Remoting Configuration](#).

18.2. CONFIGURING THE ENDPOINT



IMPORTANT

In JBoss EAP 6, the worker thread pool was configured directly in the **remoting** subsystem. In JBoss EAP 7, the remoting **endpoint** configuration references a worker from the **io** subsystem.

JBoss EAP provides the following **endpoint** configuration by default.

```
<subsystem xmlns="urn:jboss:domain:remoting:4.0">
  <endpoint/>
  ...
</subsystem>
```

Updating the Existing Endpoint Configuration

```
/subsystem=remoting/configuration=endpoint:write-attribute(name=authentication-retries,value=2)
```

```
reload
```

Creating a New Endpoint Configuration

```
/subsystem=remoting/configuration=endpoint:add
```

Deleting an Endpoint Configuration

```
/subsystem=remoting/configuration=endpoint:remove
```

```
reload
```

See [Endpoint Attributes](#) for a full list of the attributes available for the endpoint configuration.

18.3. CONFIGURING A CONNECTOR

The connector is the main configuration element relating to remoting and contains several sub-elements for additional configuration.

Updating the Existing Connector Configuration

```
/subsystem=remoting/connector=new-connector:write-attribute(name=socket-binding,value=my-socket-binding)
```

```
reload
```

Creating a New Connector

```
/subsystem=remoting/connector=new-connector:add(socket-binding=my-socket-binding)
```

Deleting a Connector

```
/subsystem=remoting/connector=new-connector:remove
```

```
reload
```

For a full list of the attributes available for configuring a connector, please see the [Remoting Subsystem Attributes](#) section.

18.4. CONFIGURING AN HTTP CONNECTOR

The HTTP connector provides the configuration for the HTTP upgrade-based remoting connector. JBoss EAP provides the following **http-connector** configuration by default.

```
<subsystem xmlns="urn:jboss:domain:remoting:4.0">
  ...
  <http-connector name="http-remoting-connector" connector-ref="default" security-
realm="ApplicationRealm"/>
</subsystem>
```

By default, this HTTP connector connects to an HTTP listener named **default** that is configured in the **undertow** subsystem. For more information, see [Configuring the Web Server \(Undertow\)](#).

Updating the Existing HTTP Connector Configuration

```
/subsystem=remoting/http-connector=new-connector:write-attribute(name=connector-ref,value=new-
connector-ref)
```

```
reload
```

Creating a New HTTP Connector

```
/subsystem=remoting/http-connector=new-connector:add(connector-ref=default)
```

Deleting an HTTP Connector

```
/subsystem=remoting/http-connector=new-connector:remove
```

See [Connector Attributes](#) for a full list of the attributes available for configuring an HTTP connector.

18.5. CONFIGURING AN OUTBOUND CONNECTION

An outbound connection is a generic remoting outbound connection that is fully specified by a URI.

Updating an Existing Outbound Connection

```
/subsystem=remoting/outbound-connection=new-outbound-connection:write-
attribute(name=uri,value=http://example.com)
```

Creating a New Outbound Connection

```
/subsystem=remoting/outbound-connection=new-outbound-connection:add(uri=http://example.com)
```

Deleting an Outbound Connection

```
/subsystem=remoting/outbound-connection=new-outbound-connection:remove
```

See [Outbound Connection Attributes](#) for a full list of the attributes available for configuring an outbound connection.

18.6. CONFIGURING A REMOTE OUTBOUND CONNECTION

A remote outbound connection is specified by a protocol, an outbound socket binding, a username and a security realm. The protocol can be either **remote**, **http-remoting** or **https-remoting**.

Updating an Existing Remote Outbound Connection

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-connection:write-attribute(name=outbound-socket-binding-ref,value=outbound-socket-binding)
```

Creating a New Remote Outbound Connection

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-connection:add(outbound-socket-binding-ref=outbound-socket-binding)
```

Deleting a Remote Outbound Connection

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-connection:remove
```

See [Remote Outbound Connection Attributes](#) for a full list of the attributes available for configuring a remote outbound connection.

18.7. CONFIGURING A LOCAL OUTBOUND CONNECTION

A local outbound connection is a remoting outbound connection with a protocol of **local**, specified only by an outbound socket binding.

Updating an Existing Local Outbound Connection

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-connection:write-attribute(name=outbound-socket-binding-ref,value=outbound-socket-binding)
```

Creating a New Local Outbound Connection

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-connection:add(outbound-socket-binding-ref=outbound-socket-binding)
```

Deleting a Local Outbound Connection

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-connection:remove
```

See [Local Outbound Connection Attributes](#) for a full list of the attributes available for configuring a local outbound connection.

18.8. ADDITIONAL REMOTING CONFIGURATION

There are several remoting elements that are configured outside of the **remoting** subsystem.

IO worker

Use the following command to set the IO worker for remoting:

```
/subsystem=remoting/configuration=endpoint:write-attribute(name=worker,
value=WORKER_NAME)
```

See [Configuring a Worker](#) for details on how to configure an IO worker.

Network interface

The network interface used by the **remoting** subsystem is the **public** interface. This interface is also used by several other subsystems, so exercise caution when modifying it.

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

In a managed domain, the **public** interface is defined per host in its **host.xml** file.

Socket binding

The default socket binding used by the **remoting** subsystem binds to port **8080**.

For more information about socket binding and socket binding groups, see [Socket Bindings](#).

Remoting connector reference for EJB

The **ejb3** subsystem contains a reference to the remoting connector for remote method invocations. The following is the default configuration:

```
<remote connector-ref="remoting-connector" thread-pool-name="default"/>
```

Secure transport configuration

Remoting transports use STARTTLS to use a secure connection, such as HTTPS, Secure Servlet, if the client requests it. The same socket binding, or network port, is used for secured and unsecured connections, so no additional server-side configuration is necessary. The client requests the secure or unsecured transport, as its needs dictate. JBoss EAP components that use remoting, such as EJBs, ORB, and the JMS provider, request secured interfaces by default.

**WARNING**

STARTTLS works by activating a secure connection if the client requests it, and otherwise defaults to an unsecured connection. It is inherently susceptible to a man-in-the-middle exploit, where an attacker intercepts the request of the client and modifies it to request an unsecured connection. Clients must be written to fail appropriately if they do not receive a secure connection, unless an unsecured connection is an appropriate fall-back.

CHAPTER 19. CONFIGURING THE IO SUBSYSTEM

19.1. IO SUBSYSTEM OVERVIEW

The **io** subsystem defines the XNIO [workers](#) and [buffer pools](#) used by other subsystems, such as Undertow and Remoting. These workers and buffer pools are defined within the following components in the **io** subsystem:

Default IO Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:io:3.0">
  <worker name="default"/>
  <buffer-pool name="default"/>
</subsystem>
```

19.2. CONFIGURING A WORKER

Workers are XNIO worker instances. An XNIO worker instance is an abstraction layer for the Java NIO APIs, which provide functionality such as management of IO and worker threads as well as SSL support. By default, JBoss EAP provides single worker called **default**, but more can be defined.

Updating an Existing Worker

To update an existing worker:

```
/subsystem=io/worker=default:write-attribute(name=io-threads,value=10)
```

```
reload
```

Creating a New Worker

To create a new worker:

```
/subsystem=io/worker=newWorker:add
```

Deleting a Worker

To delete a worker:

```
/subsystem=io/worker=newWorker:remove
```

```
reload
```

For a full list of the attributes available for configuring workers, please see the [IO Subsystem Attributes](#) section.

19.3. CONFIGURING A BUFFER POOL

**NOTE**

IO buffer pools are deprecated, but they are still set as the default in the current release. Buffer pools are pooled NIO buffer instances. Changing the buffer size has a big impact on application performance. For most servers, the ideal buffer size is usually 16k. For more information about configuring Undertow byte buffer pools, see the [Configuring Byte Buffer Pools](#) section of the *Configuration Guide* for JBoss EAP.

Updating an Existing Buffer Pool

To update an existing buffer pool:

```
/subsystem=io/buffer-pool=default:write-attribute(name=direct-buffers,value=true)
```

```
reload
```

Creating a Buffer Pool

To create a new buffer pool:

```
/subsystem=io/buffer-pool=newBuffer:add
```

Deleting a Buffer Pool

To delete a buffer pool:

```
/subsystem=io/buffer-pool=newBuffer:remove
```

```
reload
```

For a full list of the attributes available for configuring buffer pools, please see the [IO Subsystem Attributes](#) section.

19.4. TUNING THE IO SUBSYSTEM

For tips on monitoring and optimizing performance for the **io** subsystem, see the [IO Subsystem Tuning](#) section of the *Performance Tuning Guide*.

CHAPTER 20. CONFIGURING WEB SERVICES

JBoss EAP offers the ability to configure the behavior of deployed web services through the **webservices** subsystem using the management console or the management CLI. You can configure published endpoint addresses and handler chains. You can also enable the collection of runtime statistics for web services.

For more information, see [Configuring the Web Services Subsystem](#) in *Developing Web Services Applications* for JBoss EAP.

CHAPTER 21. JAKARTA SERVER FACES CONFIGURATION

21.1. MULTIPLE JAKARTA SERVER FACES IMPLEMENTATION OF JAKARTA SERVER FACES

The **jsf** subsystem allows you to install multiple Jakarta Server Faces implementations on the same JBoss EAP server instance. You can install a version of Sun Mojarra or Apache MyFaces that implements Jakarta Server Faces specification 2.3, or later.

21.1.1. Installing a Jakarta Server Faces Implementation

The following procedure describes how to manually install a new Jakarta Server Faces implementation and make it the default implementation.

1. [Add the Jakarta Server Faces implementation JAR file](#).
2. [Add the Jakarta Server Faces API JAR file](#).
3. [Add the Jakarta Server Faces injection JAR file](#).
4. Add the **commons-digester** JAR file if you are installing MyFaces.
5. [Set the default Jakarta Server Faces implementation](#).

Add the Jakarta Server Faces Implementation JAR File

1. Create the appropriate directory structure in the **EAP_HOME/modules/** directory for the Jakarta Server Faces implementation:

```
$ cd EAP_HOME/modules/
$ mkdir -p com/sun/jsf-impl/IMPL_NAME-VERSION
```



NOTE

For example, replace ***IMPL_NAME-VERSION*** with a version of Mojarra that supports the Jakarta Server Faces specification 2.3, or later.

2. Copy the Jakarta Server Faces implementation JAR file to the ***IMPL_NAME-VERSION*** subdirectory.
3. In the ***IMPL_NAME-VERSION*** subdirectory, create a **module.xml** file similar to this [Mojarra template](#) or this [MyFaces template](#). If you use a template, be sure to use appropriate values for the replaceable variables noted.

Add the Jakarta Server Faces API JAR File

1. Create the appropriate directory structure in the **EAP_HOME/modules/** directory for the Jakarta Server Faces implementation:

```
$ cd EAP_HOME/modules/
$ mkdir -p javax/faces/api/IMPL_NAME-VERSION
```

2. Copy the Jakarta Server Faces API JAR file to the ***IMPL_NAME-VERSION*** subdirectory.

3. In the ***IMPL_NAME-VERSION*** subdirectory, create a **module.xml** file similar to this [Mojarra template](#) or this [MyFaces template](#). If you use a template, be sure to use appropriate values for the replaceable variables noted.

Add the Jakarta Server Faces Injection JAR File

1. Create the appropriate directory structure in the ***EAP_HOME/modules/*** directory for the Jakarta Server Faces implementation:

```
$ cd EAP_HOME/modules/
$ mkdir -p org/jboss/as/jsf-injection/IMPL_NAME-VERSION
```

2. Follow the instructions outlined in the [Patching and Upgrading](#) guide to download the latest cumulative patch for your JBoss EAP instance. Next, complete either of the following steps:
 - If you have not applied the patch updates to the server, then copy the **wildfly-jsf-injection** and **weld-core-jsf** JAR files from ***EAP_HOME/modules/system/layers/base/org/jboss/as/jsf-injection/main/*** to the ***IMPL_NAME-VERSION*** subdirectory.
 - If you have applied the patch updates to the server, then copy the **wildfly-jsf-injection** and **weld-core-jsf** JAR files from the latest patch updates directory to the ***IMPL_NAME-VERSION*** subdirectory. For example, ***EAP_HOME/modules/system/layers/base/.overlays/layer-base-jboss-eap-7.3.z.CP/org/jboss/as/jsf-injection***, where *z* is the latest version number.
3. In the ***IMPL_NAME-VERSION*** subdirectory, create a **module.xml** file similar to this [Mojarra template](#) or this [MyFaces template](#). If you use a template, be sure to use appropriate values for the replaceable variables noted.

Add the commons-digester JAR File for MyFaces

1. Create the appropriate directory structure in the ***EAP_HOME/modules/*** directory for the **commons-digester** JAR:

```
$ cd EAP_HOME/modules/
$ mkdir -p org/apache/commons/digester/main
```

2. Download the [commons-digester](#) JAR file and copy it to the **main/** subdirectory.
3. In the **main/** subdirectory, create a **module.xml** file similar to this [template](#). If you use the template, be sure to use appropriate values for the replaceable variables noted.

Set the Default Jakarta Server Faces Implementation

1. Run the following management CLI command to set the new Jakarta Server Faces implementation as the default implementation:

```
/subsystem=jsf:write-attribute(name=default-jsf-impl-slot,value=IMPL_NAME-VERSION)
```

2. Restart the JBoss EAP server for the changes to take effect.

21.1.2. Multi-Jakarta Server Faces Implementation Support

JBoss EAP 7.3 ships with a single Jakarta Server Faces implementation, [Jakarta Server Faces 2.3](#) implementation based on Mojarrá.

Multi-Jakarta Server Faces allows the installation of multiple Jakarta Server Faces implementations and versions on the same JBoss EAP server. The goal is to allow the use of any of the Jakarta Server Faces implementations, MyFaces or Mojarra, and any version of those implementations from Java EE JSF 2.1 and beyond, and Jakarta Server Faces 2.3 and beyond. Multi-Jakarta Server Faces provides an implementation that is fully integrated with the container, which allows more efficient annotation processing, lifecycle handling, and other integration advantages.

21.1.2.1. Working of the Multi-Jakarta Server Faces Implementation

The way multi-Jakarta Server Faces works is that for each Jakarta Server Faces version, a new slot is created in the modules path under **com.sun.jsf-impl**, **javax.faces.api**, and **org.jboss.as.jsf-injection**. When the **jsf** subsystem is started, it scans the module path to find all the installed Jakarta Server Faces implementations. When the **jsf** subsystem deploys a web application containing the specified context parameter, it adds the slotted modules to the deployment.

For example, to indicate that the Jakarta Server Faces application should use MyFaces 2.2.12, assuming MyFaces 2.2.12 has been installed on the server, the following context parameter must be added:

```
<context-param>
  <param-name>org.jboss.jbossfaces.JSF_CONFIG_NAME</param-name>
  <param-value>myfaces-2.2.12</param-value>
</context-param>
```

21.1.2.2. Changing the Default Jakarta Server Faces Implementation

The multi-Jakarta Server Faces feature includes the **default-jsf-impl-slot** attribute in the **jsf** subsystem. This attribute allows you to change the default Jakarta Server Faces implementation, as described in the following procedure:

1. Use the **write-attribute** command to set the value of the **default-jsf-impl-slot** attribute to one of the active Jakarta Server Faces implementations:

```
/subsystem=jsf:write-attribute(name=default-jsf-impl-slot,value=JSF_IMPLEMENTATION)
```

2. Restart the JBoss EAP server for the change to take effect.

```
reload
```

To see which Jakarta Server Faces implementations are installed, you can execute the **list-active-jsf-impls** operation.

```
/subsystem=jsf:list-active-jsf-impls
{
  "outcome" => "success",
  "result" => [
    "myfaces-2.1.12",
    "mojarra-2.2.0-m05",
    "main"
  ]
}
```

21.2. DISALLOWING DOCTYPE DECLARATIONS

You can use the following management CLI commands to disallow **DOCTYPE** declarations in Jakarta Server Faces deployments:

```
/subsystem=jsf:write-attribute(name=disallow-doctype-decl,value=true)
reload
```

You can override this setting for a particular Jakarta Server Faces deployment by adding the **com.sun.faces.disallowDoctypeDecl** context parameter to the deployment's **web.xml** file:

```
<context-param>
  <param-name>com.sun.faces.disallowDoctypeDecl</param-name>
  <param-value>>false</param-value>
</context-param>
```


CHAPTER 22. CONFIGURING BATCH APPLICATIONS

JBoss EAP 7 introduced support for Java batch applications as defined by [JSR-352](#). The Jakarta equivalent for batch applications is [Jakarta Batch](#). You can configure an environment for running batch applications and manage batch jobs using the **batch-jberet** subsystem.

For information on developing batch applications, see [Jakarta Batch Application Development](#) in the *JBoss EAP Development Guide*.

22.1. CONFIGURING BATCH JOBS

You can configure settings for batch jobs using the **batch-jberet** subsystem, which is based on the JBeret implementation.

The default **batch-jberet** subsystem configuration defines an in-memory job repository and default thread pool settings.

```
<subsystem xmlns="urn:jboss:domain:batch-jberet:2.0">
  <default-job-repository name="in-memory"/>
  <default-thread-pool name="batch"/>
  <job-repository name="in-memory">
    <in-memory/>
  </job-repository>
  <thread-pool name="batch">
    <max-threads count="10"/>
    <keepalive-time time="30" unit="seconds"/>
  </thread-pool>
</subsystem>
```

By default, any batch jobs stopped during a server suspend will be restarted upon server resume. You can set the **restart-jobs-on-resume** property to **false** to leave jobs in the **STOPPED** state instead.

```
/subsystem=batch-jberet:write-attribute(name=restart-jobs-on-resume,value=false)
```

You can also configure the settings for batch [job repositories](#) and [thread pools](#).

22.1.1. Configure Batch Job Repositories

This section shows you how to configure in-memory and JDBC job repositories for storing batch job information using the management CLI. You can also configure job repositories using the management console by navigating to **Configuration** → **Subsystems** → **Batch (JBeret)**, clicking **View**, and selecting either **In Memory** or **JDBC** from the left-hand menu.

Add an In-memory Job Repository

You can add a job repository that stores batch job information in memory.

```
/subsystem=batch-jberet/in-memory-job-repository=REPOSITORY_NAME:add
```

Add a JDBC Job Repository

You can add a job repository that stores batch job information in a database. You must specify the name of the datasource for connecting to the database.

```
/subsystem=batch-jberet/jdbc-job-repository=REPOSITORY_NAME:add(data-source=DATASOURCE)
```

Set a Default Job Repository

You can set an in-memory or JDBC job repository as the default job repository for batch applications.

```
/subsystem=batch-jberet:write-attribute(name=default-job-repository,value=REPOSITORY_NAME)
```

This will require a server reload.

```
reload
```

22.1.2. Configure Batch Thread Pools

This section shows you how to configure thread pools and thread factories to be used for batch jobs using the management CLI. You can also configure thread pools and thread factories using the management console by navigating to **Configuration** → **Subsystems** → **Batch (JBeret)**, clicking **View**, and selecting either **Thread Factory** or **Thread Pool** from the left-hand menu.

Configure a Thread Pool

When adding a thread pool, you must specify the **max-threads**, which should always be greater than **3** as two threads are reserved to ensure partition jobs can execute as expected.

1. Add a thread pool.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:add(max-threads=10)
```

2. If desired, set a **keepalive-time** value.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:write-attribute(name=keepalive-time,value={time=60,unit=SECONDS})
```

Use a Thread Factory

1. Add a thread factory.

```
/subsystem=batch-jberet/thread-factory=THREAD_FACTORY_NAME:add
```

2. Configure the desired attributes for the thread factory.

- **group-name** - The name of a thread group to create for this thread factory.
- **priority** - The thread priority of created threads.
- **thread-name-pattern** - The template used to create names for threads. The following patterns may be used:
 - **%%** - A percent sign
 - **%t** - The per-factory thread sequence number
 - **%g** - The global thread sequence number
 - **%f** - The factory sequence number

- `%i` - The thread ID

3. Assign the thread factory to a thread pool.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:write-attribute(name=thread-factory,value=THREAD_FACTORY_NAME)
```

This will require a server reload.

```
reload
```

Set a Default Thread Pool

You can set a different thread pool as the default thread pool.

```
/subsystem=batch-jberet:write-attribute(name=default-thread-pool,value=THREAD_POOL_NAME)
```

This will require a server reload.

```
reload
```

View Thread Pool Statistics

You can view runtime information about a batch thread pool using the **read-resource** management CLI operation. You must use the **include-runtime=true** parameter in order to see this runtime information.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:read-resource(include-runtime=true)
{
  "outcome" => "success",
  "result" => {
    "active-count" => 0,
    "completed-task-count" => 0L,
    "current-thread-count" => 0,
    "keepalive-time" => undefined,
    "largest-thread-count" => 0,
    "max-threads" => 15,
    "name" => "THREAD_POOL_NAME",
    "queue-size" => 0,
    "rejected-count" => 0,
    "task-count" => 0L,
    "thread-factory" => "THREAD_FACTORY_NAME"
  }
}
```

You can also view runtime information for batch thread pools using the management console by navigating to the **Batch** subsystem from the **Runtime** tab.

22.2. MANAGING BATCH JOBS

The **batch-jberet** subsystem resource for deployments allows you to start, stop, restart, and view execution details for batch jobs. Batch jobs can be managed from the [management CLI](#) or the [management console](#).

Manage Batch Jobs from the Management CLI

Restart a Batch Job

You can restart a job that is in a **STOPPED** or **FAILED** state by providing its execution ID and optionally any properties to use when restarting the batch job.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:restart-job(execution-id=EXECUTION_ID,properties={PROPERTY=VALUE})
```

The execution ID must be the most recent execution of the job instance.

Start a Batch Job

You can start a batch job by providing the job XML file and optionally any properties to use when starting the batch job.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:start-job(job-xml-name=JOB_XML_NAME,properties={PROPERTY=VALUE})
```

Stop a Batch Job

You can stop a running batch job by providing its execution ID.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:stop-job(execution-id=EXECUTION_ID)
```

View Batch Job Execution Details

You can view the details of batch job executions. You must use the **include-runtime=true** parameter on the **read-resource** operation in order to see this runtime information.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:read-resource(recursive=true,include-runtime=true)
{
  "outcome" => "success",
  "result" => {"job" => {"import-file" => {
    "instance-count" => 2,
    "running-executions" => 0,
    "execution" => {
      "2" => {
        "batch-status" => "COMPLETED",
        "create-time" => "2016-04-11T22:03:12.708-0400",
        "end-time" => "2016-04-11T22:03:12.718-0400",
        "exit-status" => "COMPLETED",
        "instance-id" => 58L,
        "last-updated-time" => "2016-04-11T22:03:12.719-0400",
        "start-time" => "2016-04-11T22:03:12.708-0400"
      },
      "1" => {
        "batch-status" => "FAILED",
        "create-time" => "2016-04-11T21:57:17.567-0400",
        "end-time" => "2016-04-11T21:57:17.596-0400",
        "exit-status" => "Error : org.hibernate.exception.ConstraintViolationException: could not execute statement",
        "instance-id" => 15L,
        "last-updated-time" => "2016-04-11T21:57:17.597-0400",
        "start-time" => "2016-04-11T21:57:17.567-0400"
      }
    }
  }
}}
```

Manage Batch Jobs from the Management Console

To manage batch jobs from the management console, navigate to the **Runtime** tab, select the server, select **Batch (JBeret)**, and choose the job from the list.

Restart a Batch Job

Restart a **STOPPED** job by selecting the execution and clicking **Restart**.

Start a Batch Job

Start a new execution of a batch job by selecting the job and choosing **Start** from the drop down.

Stop a Batch Job

Stop a running batch job by selecting the execution and clicking **Stop**.

View Batch Job Execution Details

Job execution details are shown for each execution listed in the table.

22.3. CONFIGURE SECURITY FOR BATCH JOBS

You can configure the **batch-jberet** subsystem to run batch jobs with an Elytron security domain. This allows batch jobs to be securely suspended and resumed by the same secured identity. For example, a secured RESTful endpoint is created to initiate batch jobs using the **batch-jberet** subsystem. If both the RESTful endpoint and **batch-jberet** subsystem were secured using the same security domain, or the **batch-jberet** security domain trusted the RESTful endpoint's security domain, batch jobs initiated in this manner could be securely paused and resumed by the same secured identity.

Use the following management CLI command to update the **security-domain** attribute to configure security for batch jobs.

```
/subsystem=batch-jberet:write-attribute(name=security-domain, value=ExampleDomain)
```

```
reload
```



NOTE

Batch jobs require the **org.wildfly.extension.batch.jberet.deployment.BatchPermission** permission. It provides **start**, **stop**, **restart**, **abandon**, and **read** permissions that align with **javax.batch.operations.JobOperator**. The **default-permission-mapper** mapper provides the **org.wildfly.extension.batch.jberet.deployment.BatchPermission** permission.

CHAPTER 23. CONFIGURING THE NAMING SUBSYSTEM

23.1. ABOUT THE NAMING SUBSYSTEM

The **naming** subsystem provides the JNDI implementation for JBoss EAP. You can configure this subsystem to [bind entries in global JNDI namespaces](#). You can also configure it to [activate or deactivate the remote JNDI interface](#).

The following is an example of the **naming** subsystem XML configuration example with all of the elements and attributes specified.

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <simple name="java:global/simple-integer-binding" value="100" type="int" />
    <simple name="java:global/jboss.org/docs/url" value="https://docs.jboss.org"
type="java.net.URL" />
    <object-factory name="java:global/foo/bar/factory" module="org.foo.bar"
class="org.foo.bar.ObjectFactory" />
    <external-context name="java:global/federation/ldap/example"
class="javax.naming.directory.InitialDirContext" cache="true">
      <environment>
        <property name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory" />
        <property name="java.naming.provider.url" value="ldap://ldap.example.com:389" />
        <property name="java.naming.security.authentication" value="simple" />
        <property name="java.naming.security.principal" value="uid=admin,ou=system" />
        <property name="java.naming.security.credentials" value="secret" />
      </environment>
    </external-context>
    <lookup name="java:global/new-alias-name" lookup="java:global/original-name" />
  </bindings>
  <remote-naming/>
</subsystem>
```

23.2. CONFIGURING GLOBAL BINDINGS

The **naming** subsystem allows you to bind entries into the **java:global**, **java:jboss**, or **java** global JNDI namespaces; however, it is recommended that you use the standard portable **java:global** namespace.

Global bindings are configured in the **<bindings>** element of the **naming** subsystem. Four types of bindings are supported:

- [simple bindings](#)
- [object factory bindings](#)
- [external context bindings](#)
- [binding lookup aliases](#)

Configuring Simple Bindings

The **simple** XML configuration element binds primitive or **java.net.URL** entries.

- The **name** attribute is mandatory and specifies the target JNDI name for the entry.

- The **value** attribute is mandatory and defines the entry's value.
- The optional **type** attribute, which defaults to **java.lang.String**, specifies the type of the entry's value. In addition to **java.lang.String**, you can specify primitive types and their corresponding object wrapper classes, such as **int** or **java.lang.Integer**, and **java.net.URL**.

The following is an example of a management CLI command that creates a simple binding.

```
/subsystem=naming/binding=java\:global/simple-integer-binding:add(binding-type=simple, type=int, value=100)
```

Resulting XML Configuration

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <simple name="java\:global/simple-integer-binding" value="100" type="int"/>
  </bindings>
  <remote-naming/>
</subsystem>
```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global/simple-integer-binding:remove
```

Binding Object Factories

The **object-factory** XML configuration element binds **javax.naming.spi.ObjectFactory** entries.

- The **name** attribute is mandatory and specifies the target JNDI name for the entry.
- The **class** attribute is mandatory and defines the object factory's Java type.
- The **module** attribute is mandatory and specifies the JBoss Module ID where the object factory Java class can be loaded from.
- The optional **environment** child element can be used to provide a custom environment to the object factory.

The following is an example of a management CLI command that creates an object factory binding.

```
/subsystem=naming/binding=java\:global/foo/bar/factory:add(binding-type=object-factory, module=org.foo.bar, class=org.foo.bar.ObjectFactory, environment=[p1=v1, p2=v2])
```

Resulting XML Configuration

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <object-factory name="java\:global/foo/bar/factory" module="org.foo.bar"
class="org.foo.bar.ObjectFactory">
      <environment>
        <property name="p1" value="v1" />
        <property name="p2" value="v2" />
      </environment>
    </object-factory>
  </bindings>
</subsystem>
```

```

    </object-factory>
  </bindings>
</subsystem>

```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global\foo\bar\factory:remove
```

Binding External Contexts

Federation of external JNDI contexts, such as LDAP context, are achieved using the **external-context** XML configuration element.

- The **name** attribute is mandatory and specifies the target JNDI name for the entry.
- The **class** attribute is mandatory and indicates the Java initial naming context type used to create the federated context. Note that such type must have a constructor with a single environment map argument.
- The optional **module** attribute specifies the JBoss Module ID where any classes required by the external JNDI context can be loaded from.
- The optional **cache** attribute, which defaults to **false**, indicates whether the external context instance should be cached.
- The optional **environment** child element can be used to provide the custom environment needed to look up the external context.

The following is an example of a management CLI command that creates an external context binding.

```
/subsystem=naming/binding=java\:global\federation\ldap\example:add(binding-type=external-
context, cache=true, class=javax.naming.directory.InitialDirContext, module=org.jboss.as.naming,
environment=[java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory,
java.naming.provider.url="ldap://ldap.example.com:389", java.naming.security.authentication=simple,
java.naming.security.principal="uid=admin,ou=system", java.naming.security.credentials=secret])
```

Resulting XML Configuration

```

<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <external-context name="java:global/federation/ldap/example" module="org.jboss.as.naming"
class="javax.naming.directory.InitialDirContext" cache="true">
      <environment>
        <property name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory"/>
        <property name="java.naming.provider.url" value="ldap://ldap.example.com:389"/>
        <property name="java.naming.security.authentication" value="simple"/>
        <property name="java.naming.security.principal" value="uid=admin,ou=system"/>
        <property name="java.naming.security.credentials" value="secret"/>
      </environment>
    </external-context>
  </bindings>
</subsystem>

```

Use the following command to remove the binding.


```
/subsystem=naming/binding=java\:global\federation\ldap\example:remove
```

NOTE

Resource look up for JNDI providers that do not properly implement the **lookup(Name)** method can result in a "javax.naming.InvalidNameException: Only support CompoundName names" error.

You might be able to work around this issue by specifying that the external context environment use the **lookup(String)** method instead by adding the following property; however, this can result in a performance degradation.

```
<property name="org.jboss.as.naming.lookup.by.string" value="true"/>
```

Binding Lookup Aliases

The **lookup** element allows you to bind existing entries into additional names or aliases.

- The **name** attribute is mandatory and specifies the target JNDI name for the entry.
- The **lookup** attribute is mandatory and indicates the source JNDI name.

The following is an example of a management CLI command that binds an existing entry to an alias.

```
/subsystem=naming/binding=java\:global\new-alias-name:add(binding-type=lookup,
lookup=java\:global\original-name)
```

Resulting XML Configuration

```
<lookup name="java\:global/new-alias-name" lookup="java\:global/original-name" />
```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global\c:remove
```

23.3. DYNAMICALLY CHANGE JNDI BINDINGS

JBoss EAP 7.1 introduced the ability to dynamically change JNDI bindings without forcing a server reload or restart. This feature can be useful if the network service endpoints are dynamically reconfigured due to version updates, testing requirements, or application feature updates.

To update a JNDI binding, use the **rebind** operation. The **rebind** operation takes the same arguments as the **add** operation. This command works for all binding types except for **external-context** binding types. External context bindings require additional dependencies, which affect the Modular Service Container (MSC) state, so they cannot be restarted without restarting services.

The following command dynamically changes the JNDI binding that was defined in the [Configuring Simple Bindings](#) example.

```
/subsystem=naming/binding=java\:global\simple-integer-binding:rebind(binding-type=simple,
type=int, value=200)
```

For more information about how to configure global bindings in the **naming** subsystem, see [Configuring Global Bindings](#).

23.4. CONFIGURING THE REMOTE JNDI INTERFACE

The remote JNDI interface allows clients to look up entries in remote JBoss EAP instances. The **naming** subsystem can be configured to deactivate or activate this interface, which is activated by default. The remote JNDI interface is configured using the **<remote-naming>** element.

Use the following management CLI command to activate or reactivate the remote JNDI interface.

```
/subsystem=naming/service=remote-naming:add
```

Use the following management CLI command to deactivate the remote JNDI interface.

```
/subsystem=naming/service=remote-naming:remove
```



NOTE

Only entries within the **java:jboss/exported** context are accessible over remote JNDI.

CHAPTER 24. CONFIGURING HIGH AVAILABILITY

24.1. INTRODUCTION TO HIGH AVAILABILITY

JBoss EAP provides the following *high availability* services to guarantee the availability of deployed Jakarta EE applications.

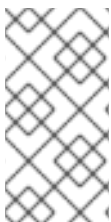
Load balancing

This allows a service to handle a large number of requests by spreading the workload across multiple servers. A client can have timely responses from the service even in the event of a high volume of requests.

Failover

This allows a client to have uninterrupted access to a service even in the event of hardware or network failures. If the service fails, another cluster member takes over the client's requests so that it can continue processing.

Clustering is a term that encompasses all of these capabilities. Members of a cluster can be configured to share workloads, referred to as load balancing, and pick up client processing in the event of a failure of another cluster member, referred to as failover.



NOTE

It is important to keep in mind that the JBoss EAP operating mode chosen, either *standalone server* or *managed domain*, pertains to how you want to manage your servers. High availability services can be configured in JBoss EAP regardless of its operating mode.

JBoss EAP supports high availability at several different levels using various components. Some of those components of the runtime and your applications that can be made highly-available are:

- Instances of the application server
- Web applications, when used in conjunction with the internal JBoss Web Server, Apache HTTP Server, Microsoft IIS, or Oracle iPlanet Web Server
- Stateful and stateless session Enterprise JavaBeans (EJBs)
- Single sign-on (SSO) mechanisms
- HTTP sessions
- JMS services and message-driven beans (MDBs)
- Singleton MSC services
- Singleton deployments

Clustering is made available to JBoss EAP by the [jgroups](#), [infinispan](#), and [modcluster](#) subsystems. The *ha* and *full-ha* profiles have these systems enabled. In JBoss EAP, these services start up and shut down on demand, but they will only start up if an application configured as *distributable* is deployed on the servers.

See the JBoss EAP *Development Guide* for how to [mark an application as distributable](#).

24.2. CLUSTER COMMUNICATION WITH JGROUPS

24.2.1. About JGroups

JGroups is a toolkit for reliable messaging and can be used to create clusters whose nodes can send messages to each other.

The **jgroups** subsystem provides group communication support for high availability services in JBoss EAP. It allows you to configure named channels and protocol stacks as well as view runtime statistics for channels. The **jgroups** subsystem is available when using a configuration that provides high availability capabilities, such as the *ha* or *full-ha* profile in a managed domain, or the **standalone-ha.xml** or **standalone-full-ha.xml** configuration file for a standalone server.

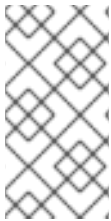
JBoss EAP is preconfigured with two JGroups stacks:

udp

The nodes in the cluster use User Datagram Protocol (UDP) multicasting to communicate with each other. This is the default stack.

tcp

The nodes in the cluster use Transmission Control Protocol (TCP) to communicate with each other.



NOTE

TCP has more overhead and is often considered slower than UDP since it handles error checking, packet ordering, and congestion control itself. JGroups handles these features for UDP, whereas TCP guarantees them itself. TCP is a good choice when using JGroups on unreliable or high congestion networks, or when multicast is not available.

You can use the preconfigured stacks or define your own to suit your system's specific requirements. For additional information on the available protocols and their attributes, see the following sections.

- [JGroups Subsystem Attributes](#)
- [JGroups Protocols](#)

24.2.2. Switch the Default JGroups Channel to Use TCP

By default, cluster nodes communicate using the UDP protocol. The default **ee** JGroups channel uses the predefined **udp** protocol stack.

```
<channels default="ee">
  <channel name="ee" stack="udp"/>
</channels>
<stacks>
  <stack name="udp">
    <transport type="UDP" socket-binding="jgroups-udp"/>
    <protocol type="PING"/>
    ...
  </stack>
  <stack name="tcp">
    <transport type="TCP" socket-binding="jgroups-tcp"/>
    <protocol type="MPING" socket-binding="jgroups-mping"/>
  </stack>
```

```
...
</stack>
</stacks>
```

Some networks only allow TCP to be used. Use the following management CLI command to switch the **ee** channel to use the preconfigured **tcp** stack.

```
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcp)
```

This default **tcp** stack uses the **MPING** protocol, which uses IP multicast to discover the initial cluster membership. See the following sections for configuring stacks for alternative membership discovery protocols:

- Using the [TCPPING](#) protocol to define a static cluster membership list.
- Using the [TCPGOSSIP](#) protocol to use an external gossip router to discover the members of a cluster.
- Using the [JDBC_PING](#) protocol to use a database to discover members of a cluster.

24.2.3. Configure TCPPING

This procedure creates a new JGroups stack that uses the **TCPPING** protocol to define a static cluster membership list. A base script is provided that creates a **tcpping** stack and sets the default **ee** channel to use this new stack. The management CLI commands in this script must be customized for your environment and will be processed as a batch.

1. Copy the following script into a text editor and save it to the local file system.

```
# Define the socket bindings
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=jgroups-host-a:add(host=HOST_A,port=7600)
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=jgroups-host-b:add(host=HOST_B,port=7600)
batch
# Add the tcpping stack
/subsystem=jgroups/stack=tcpping:add
/subsystem=jgroups/stack=tcpping/transport=TCP:add(socket-binding=jgroups-tcp)
/subsystem=jgroups/stack=tcpping/protocol=TCPPING:add(socket-bindings=[jgroups-host-
a,jgroups-host-b])
/subsystem=jgroups/stack=tcpping/protocol=MERGE3:add
/subsystem=jgroups/stack=tcpping/protocol=FD_SOCK:add
/subsystem=jgroups/stack=tcpping/protocol=FD_ALL:add
/subsystem=jgroups/stack=tcpping/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=tcpping/protocol=UNICAST3:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=tcpping/protocol=MFC:add
/subsystem=jgroups/stack=tcpping/protocol=FRAG2:add
# Set tcpping as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcpping)
run-batch
reload
```

Note that the order of protocols defined is important. You can also insert a protocol at a particular index by passing an **add-index** value to the **add** command. The index is zero-based, so the following management CLI command adds the **UNICAST3** protocol as the seventh protocol.

```
/subsystem=jgroups/stack=tcpping/protocol=UNICAST3:add(add-index=6)
```

2. Modify the script for your environment.

- If you are running in a managed domain, you must specify which profile to update by preceding the **/subsystem=jgroups** commands with **/profile=PROFILE_NAME**.
- Adjust the following properties as appropriate for your environment:
 - **socket-bindings**: A comma-separated list of the host and port combinations that are considered well-known and will be available to look up the initial membership. See [Configuring Socket Bindings](#) for more information on defining socket bindings.
 - **initial_hosts**: A comma-separated list of the host and port combinations, using the syntax **HOST[PORT]**, that are considered well-known and will be available to look up the initial membership, for example, **host1[1000],host2[2000]**.
 - **port_range**: This property is used to extend the **initial_hosts** port range by the specified value. For example, if you set the **initial_hosts** to **host1[1000],host2[2000]**, and the **port_range** to **1**, the **initial_hosts** setting expands to **host1[1000],host1[1001],host2[2000],host2[2001]**. This property only works with the **initial_hosts** property.

3. Run the script by passing the script file to the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

The TCPPING stack is now available and TCP is used for network communication.

24.2.4. Configure TCPGOSSIP

This procedure creates a new JGroups stack that uses the **TCPGOSSIP** protocol to use an external gossip router to discover the members of a cluster. A base script is provided that creates a **tcpgossip** stack and sets the default **ee** channel to use this new stack. The management CLI commands in this script must be customized for your environment and will be processed as a batch.

1. Copy the following script into a text editor and save it to the local file system.

```
# Define the socket bindings
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=jgroups-host-a:add(host=HOST_A,port=13001)
batch
# Add the tcpgossip stack
/subsystem=jgroups/stack=tcpgossip:add
/subsystem=jgroups/stack=tcpgossip/transport=TCP:add(socket-binding=jgroups-tcp)
/subsystem=jgroups/stack=tcpgossip/protocol=TCPGOSSIP:add(socket-bindings=[jgroups-
host-a])
/subsystem=jgroups/stack=tcpgossip/protocol=MERGE3:add
/subsystem=jgroups/stack=tcpgossip/protocol=FD SOCK:add
/subsystem=jgroups/stack=tcpgossip/protocol=FD_ALL:add
```

```

/subsystem=jgroups/stack=tcpgossip/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=tcpgossip/protocol=UNICAST3:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=tcpgossip/protocol=MFC:add
/subsystem=jgroups/stack=tcpgossip/protocol=FRAG2:add
# Set tcpgossip as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcpgossip)
run-batch
reload

```

Note that the order of protocols defined is important. You can also insert a protocol at a particular index by passing an **add-index** value to the **add** command. The index is zero-based, so the following management CLI command adds the **UNICAST3** protocol as the seventh protocol.

```

/subsystem=jgroups/stack=tcpgossip/protocol=UNICAST3:add(add-index=6)

```

2. Modify the script for your environment.

- If you are running in a managed domain, you must specify which profile to update by preceding the **/subsystem=jgroups** commands with **/profile=PROFILE_NAME**.
- Adjust the following properties as appropriate for your environment:
 - **socket-bindings**: A comma-separated list of the host and port combinations that are considered well-known and will be available to look up the initial membership. See [Configuring Socket Bindings](#) for more information on defining socket bindings.
 - **initial_hosts**: A comma-separated list of the host and port combinations, using the syntax **HOST[PORT]**, that are considered well-known and will be available to look up the initial membership, for example, **host1[1000],host2[2000]**.
 - **port_range**: This property is used to extend the **initial_hosts** port range by the specified value. For example, if you set the **initial_hosts** to **host1[1000],host2[2000]**, and the **port_range** to **1**, the **initial_hosts** setting expands to **host1[1000],host1[1001],host2[2000],host2[2001]**. This property only works with the **initial_hosts** property.
 - **reconnect_interval**: The interval in milliseconds by which a disconnected stub attempts to reconnect to the gossip router.
 - **sock_conn_timeout**: The maximum time for socket creation. The default is **1000** milliseconds.
 - **sock_read_timeout**: The maximum time in milliseconds to block on a read. A value of **0** will block indefinitely.

3. Run the script by passing the script file to the management CLI.

```

$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME

```

The TCPGOSSIP stack is now available and TCP is used for network communication. This stack is configured for use with a gossip router so that JGroups cluster members can find other cluster members.

24.2.5. Configure JDBC_PING

You can use the **JDBC_PING** protocol to manage and discover membership in a cluster.

JDBC_PING uses a database, specified in a data-source, to list the members of the cluster.

1. Create a data-source to connect to the database you want to use to manage cluster membership.
2. Copy the following script into a text editor and save it to the local file system.

```
batch
# Add the JDBC_PING stack
/subsystem=jgroups/stack=JDBC_PING:add
/subsystem=jgroups/stack=JDBC_PING/transport=TCP:add(socket-binding=jgroups-tcp)
/subsystem=jgroups/stack=JDBC_PING/protocol=JDBC_PING:add(data-
source=ExampleDS)
/subsystem=jgroups/stack=JDBC_PING/protocol=MERGE3:add
/subsystem=jgroups/stack=JDBC_PING/protocol=FD SOCK:add
/subsystem=jgroups/stack=JDBC_PING/protocol=FD_ALL:add
/subsystem=jgroups/stack=JDBC_PING/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=JDBC_PING/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=JDBC_PING/protocol=UNICAST3:add
/subsystem=jgroups/stack=JDBC_PING/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=JDBC_PING/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=JDBC_PING/protocol=MFC:add
/subsystem=jgroups/stack=JDBC_PING/protocol=FRAG2:add
# Set JDBC_PING as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=JDBC_PING)
run-batch
reload
```

Note that the order of protocols defined is important. You can also insert a protocol at a particular index by passing an **add-index** value to the **add** command. The index is zero-based, so the following management CLI command adds the **UNICAST3** protocol as the seventh protocol.

```
/subsystem=jgroups/stack=JDBC_PING/protocol=UNICAST3:add(add-index=6)
```

3. Modify the script for your environment.
 - If you are running in a managed domain, you must specify which profile to update by preceding the **/subsystem=jgroups** commands with **/profile=PROFILE_NAME**.
 - Replace 'ExampleDS' with the name of the data-source you defined in Step 1.
4. Run the script by passing the script file to the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

The JDBC_PING stack is now available and TCP is used for network communication.

Related information

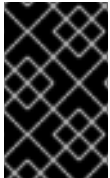
JDBC_PING: <https://developer.jboss.org/wiki/JDBCPING>

Creating a data-source: [About JBoss EAP Datasources](#)

24.2.6. Binding JGroups to a Network Interface

By default, JGroups only binds to the **private** network interface, which points to localhost in the default configuration. For security reasons, JGroups will not bind to the network interface defined by the **-b** argument specified during JBoss EAP startup, as clustering traffic should not be exposed on a public network interface.

See the [Network and Port Configuration](#) chapter in this guide for information about how to configure network interfaces.



IMPORTANT

For security reasons, JGroups should only be bound to a non-public network interface. For performance reasons, we also recommend that the network interface for JGroups traffic should be part of a dedicated Virtual Local Area Network (VLAN).

24.2.7. Securing a Cluster

There are several concerns to address in order to run a cluster securely:

- Preventing unauthorized nodes from joining the cluster. This is addressed by requiring [authentication](#).
- Preventing non-members from communicating with cluster members. This is addressed by [encrypting messages](#).

24.2.7.1. Configuring Authentication

JGroups authentication is performed by the **AUTH** protocol. The purpose is to ensure that only authenticated nodes can join a cluster.

In the applicable server configuration file, add the **AUTH** protocol with the appropriate property settings. The **AUTH** protocol should be configured immediately before the **pbcast.GMS** protocol.

The following examples demonstrate using **AUTH** with different forms of authorization tokens.

AUTH with a Simple Token

```
...
<protocol type="pbcast.STABLE"/>
<auth-protocol type="AUTH">
  <plain-token>
    <shared-secret-reference clear-text="my_password"/>
  </plain-token>
</auth-protocol>
<protocol type="pbcast.GMS"/>
...
```

AUTH with a Digest Algorithm Token

This format can be used with any digest algorithm, for example MD5 or SHA-2. The default digest algorithm in JBoss EAP 7.3 is SHA-256, the strongest digest algorithm required to be supported by the JVM. Many JVMs will additionally implement SHA-512.

```
...
<protocol type="pbcast.STABLE"/>
```

```

<auth-protocol type="AUTH">
  <digest-token algorithm="SHA-512">
    <shared-secret-reference clear-text="my_password"/>
  </digest-token>
</auth-protocol>
<protocol type="pbcast.GMS"/>
...

```

AUTH with an X509 Token

This example creates a new keystore in the **elytron** subsystem, and references it in the JGroups **AUTH** configuration.

1. Create the keystore:

```
$ keytool -genkeypair -alias jgroups_key -keypass my_password -storepass my_password -storetype jks -keystore jgroups.keystore -keyalg RSA
```

2. Add the keystore to the **elytron** subsystem using the management CLI:

```
/subsystem=elytron/key-store=jgroups-token-store:add(type=jks,path=/path/to/jgroups.keystore,credential-reference={clear-text=my_password}, required=true)
```

3. In the JGroups stack definition, configure **AUTH** to use the keystore:

```

...
<protocol type="pbcast.STABLE"/>
<auth-protocol type="AUTH">
  <cipher-token algorithm="RSA" key-alias="jgroups_key" key-store="jgroups-token-store">
    <shared-secret-reference clear-text="my_password"/>
    <key-credential-reference clear-text="my_password"/>
  </cipher-token>
</auth-protocol>
<protocol type="pbcast.GMS"/>
...

```

24.2.7.2. Configuring Encryption

To encrypt messages, JGroups uses a secret key that is shared by the members of a cluster. The sender encrypts the message using the shared secret key, and the receiver decrypts the message using the same secret key. With [symmetric encryption](#), which is configured using the **SYM_ENCRYPT** protocol, nodes use a shared keystore to retrieve the secret key. With [asymmetric encryption](#), which is configured using the **ASYM_ENCRYPT** protocol, nodes retrieve the secret key from the coordinator of the cluster after being authenticated using **AUTH**.

Using Symmetric Encryption

In order to use **SYM_ENCRYPT**, you must set up a keystore that will be referenced in the JGroups configuration for each node.

1. Create a keystore.

In the following command, replace **VERSION** with the appropriate JGroups JAR version and **PASSWORD** with a keystore password.

```
$ java -cp EAP_HOME/modules/system/layers/base/org/jgroups/main/jgroups-VERSION.jar
org.jgroups.demos.KeyStoreGenerator --alg AES --size 128 --storeName
defaultStore.keystore --storepass PASSWORD --alias mykey
```

This will generate a **defaultStore.keystore** file that will be referenced in the JGroups configuration.

2. Once the keystore has been generated it is defined in the **SYM_PROTOCOL** using one of two methods.
 - Specify the keystore [directly in the configuration](#).
 - Reference the keystore [using the Elytron subsystem](#).



NOTE

Configuring **AUTH** is optional when using **SYM_ENCRYPT**.

Using Symmetric Encryption by Directly Referencing the Keystore

1. Configure the **SYM_ENCRYPT** protocol in the **jgroups** subsystem.
In the applicable server configuration file, add the **SYM_ENCRYPT** protocol with the appropriate property settings. This protocol will reference the previously created keystore. The **SYM_ENCRYPT** protocol should be configured immediately before the **pbcast.NAKACK2** protocol.

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="SYM_ENCRYPT">
        <property name="provider">SunJCE</property>
        <property name="sym_algorithm">AES</property>
        <property name="encrypt_entire_message">true</property>
        <property name="keystore_name">/path/to/defaultStore.keystore</property>
        <property name="store_password">PASSWORD</property>
        <property name="alias">mykey</property>
      </protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

Using Symmetric Encryption with Elytron

1. Using the management CLI, create a keystore in the **elytron** subsystem that references the **defaultStore.keystore** created in [using symmetric encryption](#).

```
/subsystem=elytron/key-store=jgroups-
keystore:add(path=/path/to/defaultStore.keystore,credential-reference={clear-
text=PASSWORD},type=JCEKS)
```

2. Add the **SYM_ENCRYPT** protocol in the **jgroups** subsystem with the appropriate property settings. The **SYM_ENCRYPT** protocol should be configured immediately before the **pbcast.NAKACK2** protocol, as seen in the following configuration.

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <encrypt-protocol type="SYM_ENCRYPT" key-alias="mykey" key-store="jgroups-
keystore">
        <key-credential-reference clear-text="PASSWORD"/>
        <property name="provider">SunJCE</property>
        <property name="encrypt_entire_message">true</property>
      </encrypt-protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```



NOTE

The above example uses a clear text password; however, a credential store may also be defined to define the password outside of the configuration file. For more information on configuring this store, see the [Credential Store](#) section in the *How to Configure Server Security* guide.

Using Asymmetric Encryption

In order to use **ASYM_ENCRYPT** the **AUTH** protocol must be defined. See the [Configuring Authentication](#) section for instructions on configuring the **AUTH** protocol in the **jgroups** subsystem.

The **ASYM_ENCRYPT** is configured using one of two methods.

- Generate a secret key and reference it [directly in the configuration](#).
- Create a keystore and reference it [using the Elytron subsystem](#).

Using Asymmetric Encryption by Generating a Secret Key

1. Configure the **ASYM_ENCRYPT** protocol in the **jgroups** subsystem.
In the applicable server configuration file, add the **ASYM_ENCRYPT** protocol with the appropriate property settings. The **ASYM_ENCRYPT** protocol should be configured immediately before the **pbcast.NAKACK2** protocol, as seen in the following configuration.

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="ASYM_ENCRYPT">
        <property name="encrypt_entire_message">true</property>
        <property name="sym_keylength">128</property>
        <property name="sym_algorithm">AES/ECB/PKCS5Padding</property>
        <property name="asym_keylength">512</property>
        <property name="asym_algorithm">RSA</property>
      </protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <!-- Configure AUTH protocol here -->
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

Using Asymmetric Encryption with Elytron

1. Create a keystore to contain the key pair. The following command creates a keystore with the **mykey** entry.

```
$ keytool -genkeypair -alias mykey -keyalg RSA -keysize 1024 -keystore
defaultKeystore.keystore -dname "CN=localhost" -keypass secret -storepass secret
```

2. Using the management CLI, create a keystore in the **elytron** subsystem that references the **defaultStore.keystore**.

```
/subsystem=elytron/key-store=jgroups-
keystore:add(path=/path/to/defaultStore.keystore,credential-reference={clear-
text=PASSWORD},type=JCEKS)
```

3. Add the **ASYM_ENCRYPT** protocol in the **jgroups** subsystem with the appropriate property settings. The **ASYM_ENCRYPT** protocol should be configured immediately before the **pbcast.NAKACK2** protocol, as seen in the following configuration.

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
```

```

<transport type="UDP" socket-binding="jgroups-udp"/>
<protocol type="PING"/>
<protocol type="MERGE3"/>
<protocol type="FD SOCK"/>
<protocol type="FD_ALL"/>
<protocol type="VERIFY_SUSPECT"/>
<encrypt-protocol type="ASYM_ENCRYPT" key-alias="mykey" key-store="jgroups-
keystore">
  <key-credential-reference clear-text="secret" />
  <property name="encrypt_entire_message">true</property>
</encrypt-protocol>
<protocol type="pbcast.NAKACK2"/>
<protocol type="UNICAST3"/>
<protocol type="pbcast.STABLE"/>
<!-- Configure AUTH protocol here -->
<protocol type="pbcast.GMS"/>
<protocol type="UFC"/>
<protocol type="MFC"/>
<protocol type="FRAG2"/>
</stack>
</stacks>
</subsystem>

```



NOTE

The above example uses a clear text password; however, a credential store may also be defined to define the password outside of the configuration file. For more information on configuring this store, see the [Credential Store](#) section in the *How to Configure Server Security* guide.

24.2.8. Configure JGroups Thread Pools

The **jgroups** subsystem contains the **default**, **internal**, **oob**, and **timer** thread pools. These pools can be configured for any JGroups stack.

The following table lists the attributes you can configure for each thread pool and the default value for each.

Thread Pool Name	keepalive-time	max-threads	min-threads	queue-length
default	60000L	300	20	100
internal	60000L	4	2	100
oob	60000L	300	20	0
timer	5000L	4	2	500

Use the following syntax to configure a JGroups thread pool using the management CLI.

```
/subsystem=jgroups/stack=STACK_TYPE/transport=TRANSPORT_TYPE/thread-
pool=THREAD_POOL_NAME:write-attribute(name=ATTRIBUTE_NAME,
value=ATTRIBUTE_VALUE)
```

The following is an example of the management CLI command to set the **max-threads** value to **500** in the **default** thread pool for the **udp** stack.

```
/subsystem=jgroups/stack=udp/transport=UDP/thread-pool=default:write-attribute(name="max-
threads", value="500")
```

24.2.9. Configure JGroups Send and Receive Buffers

Resolving Buffer Size Warnings

By default, JGroups is configured with certain send and receive buffer values; however, your operating system may limit the available buffer sizes and JBoss EAP may not be able to use its configured buffer values. In this situation you will see warnings in the JBoss EAP logs similar to the following:

```
WARNING [org.jgroups.protocols.UDP] (ServerService Thread Pool -- 68)
JGRP000015: the send buffer of socket DatagramSocket was set to 640KB, but the OS only
allocated 212.99KB.
This might lead to performance problems. Please set your max send buffer in the OS correctly (e.g.
net.core.wmem_max on Linux)
WARNING [org.jgroups.protocols.UDP] (ServerService Thread Pool -- 68)
JGRP000015: the receive buffer of socket DatagramSocket was set to 20MB, but the OS only
allocated 212.99KB.
This might lead to performance problems. Please set your max receive buffer in the OS correctly
(e.g. net.core.rmem_max on Linux)
```

To resolve this, consult your operating system documentation for instructions on how to increase the buffer size. For Red Hat Enterprise Linux systems, edit **/etc/sysctl.conf** as the root user to configure maximum values for buffer sizes that will survive system restarts. For example:

```
# Allow a 25MB UDP receive buffer for JGroups
net.core.rmem_max = 26214400
# Allow a 1MB UDP send buffer for JGroups
net.core.wmem_max = 1048576
```

After modifying **/etc/sysctl.conf**, run **sysctl -p** for the changes to take effect.

Configuring JGroups Buffer Sizes

You can configure the JGroups buffer sizes that JBoss EAP uses by setting the following transport properties on the UDP and TCP JGroups stacks.

UDP Stack

- **ucast_rcv_buf_size**
- **ucast_send_buf_size**
- **mcast_rcv_buf_size**
- **mcast_send_buf_size**

TCP Stack

- **recv_buf_size**
- **send_buf_size**

JGroups buffer sizes can be configured using the management console or the management CLI.

Use the following syntax to set a JGroups buffer size property using the management CLI.

```
/subsystem=jgroups/stack=STACK_NAME/transport=TRANSPORT/property=PROPERTY_NAME:add(value=BUFFER_SIZE)
```

The following is an example management CLI command to set the **recv_buf_size** property to **20000000** on the **tcp** stack.

```
/subsystem=jgroups/stack=tcp/transport=TRANSPORT/property=recv_buf_size:add(value=20000000)
```

JGroups buffer sizes can also be configured using the management console by navigating to the **JGroups** subsystem from the **Configuration** tab, clicking **View**, selecting the **Stack** tab, choosing the appropriate stack, clicking **Transport**, and editing the **Properties** field.

24.2.10. Tuning the JGroups Subsystem

For tips on monitoring and optimizing performance for the **jgroups** subsystem, see the [JGroups Subsystem Tuning](#) section of the *Performance Tuning Guide*.

24.2.11. JGroups Troubleshooting

24.2.11.1. Nodes Do Not Form a Cluster

Make sure your machine is set up correctly for IP multicast. There are two test programs that ship with JBoss EAP that can be used to test IP multicast: **McastReceiverTest** and **McastSenderTest**.

In a terminal, start **McastReceiverTest**.

```
$ java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastReceiverTest -mcast_addr 230.11.11.11 -port 5555
```

Then in another terminal window, start **McastSenderTest**.

```
$ java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastSenderTest -mcast_addr 230.11.11.11 -port 5555
```

If you want to bind to a specific network interface card (NIC), use **-bind_addr YOUR_BIND_ADDRESS**, where **YOUR_BIND_ADDRESS** is the IP address of the NIC to which you want to bind. Use this parameter in both the sender and the receiver.

When you type in the **McastSenderTest** terminal window, you should see the output in the **McastReceiverTest** window. If you do not, try the following steps.

- Increase the time-to-live for multicast packets by adding **-ttl VALUE** to the sender command. The default used by this test program is **32** and the **VALUE** must be no greater than **255**.
- If the machines have multiple interfaces, verify that you are using the correct interface.

- Contact a system administrator to make sure that multicast will work on the interface you have chosen.

Once you know multicast is working properly on each machine in your cluster, you can repeat the above test to test the network, putting the sender on one machine and the receiver on another.

24.2.11.2. Causes of Missing Heartbeats in Failure Detection

Sometimes a cluster member is suspected by failure detection (FD) because a heartbeat acknowledgement has not been received for some time **T**, which is defined by **timeout** and **max_tries**.

For an example cluster of nodes A, B, C, and D, where A pings B, B pings C, C pings D, and D pings A, C can be suspected for any of the following reasons:

- B or C are running at 100% CPU for more than **T** seconds. So even if C sends a heartbeat acknowledgement to B, B may not be able to process it because it is at 100% CPU usage.
- B or C are garbage collecting, which results in the same situation as above.
- A combination of the two cases above.
- The network loses packets. This usually happens when there is a lot of traffic on the network, and the switch starts dropping packets, usually broadcasts first, then IP multicasts, and TCP packets last.
- B or C are processing a callback. For example, if C received a remote method call over its channel that takes **T** + 1 seconds to process, during this time C will not process any other messages, including heartbeats. Therefore, B will not receive the heartbeat acknowledgement and will suspect C.

24.3. INFINISPAN

24.3.1. About Infinispan

Infinispan is a Java data grid platform that provides a [JSR-107](#)-compatible cache interface for managing cached data. The Jakarta equivalent of managing cached data is the [Jakarta Persistence 2.2](#).

For more information about Infinispan functionality and configuration options see the [Infinispan Documentation](#).

The **infinispan** subsystem provides caching support for JBoss EAP. It allows you to configure and view runtime metrics for named cache containers and caches.

When using a configuration that provides high availability capabilities, such as the *ha* or *full-ha* profile in a managed domain, or the **standalone-ha.xml** or **standalone-full-ha.xml** configuration file for a standalone server, the **infinispan** subsystem provides caching, state replication, and state distribution support. In non-high-availability configurations, the **infinispan** subsystem provides local caching support.



IMPORTANT

Infinispan is delivered as a private module in JBoss EAP to provide the caching capabilities of JBoss EAP. Infinispan is not supported for direct use by applications.

24.3.2. Cache Containers

A cache container is a repository for the caches used by a subsystem. Each cache container defines a default cache to be used.

JBoss EAP 7 defines the following default Infinispan cache containers:

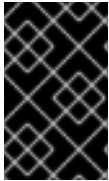
- **server** for singleton caching
- **web** for web session clustering
- **ejb** for stateful session bean clustering
- **hibernate** for entity caching

Example: Default Infinispan Configuration

```
<subsystem xmlns="urn:jboss:domain:infinispan:7.0">
  <cache-container name="server" aliases="singleton cluster" default-cache="default"
module="org.wildfly.clustering.server">
    <transport lock-timeout="60000"/>
    <replicated-cache name="default">
      <transaction mode="BATCH"/>
    </replicated-cache>
  </cache-container>
  <cache-container name="web" default-cache="dist" module="org.wildfly.clustering.web.infinispan">
    <transport lock-timeout="60000"/>
    <distributed-cache name="dist">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <file-store/>
    </distributed-cache>
  </cache-container>
  <cache-container name="ejb" aliases="sfsb" default-cache="dist"
module="org.wildfly.clustering.ejb.infinispan">
    <transport lock-timeout="60000"/>
    <distributed-cache name="dist">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <file-store/>
    </distributed-cache>
  </cache-container>
  <cache-container name="hibernate" default-cache="local-query" module="org.hibernate.infinispan">
    <transport lock-timeout="60000"/>
    <local-cache name="local-query">
      <object-memory size="1000"/>
      <expiration max-idle="100000"/>
    </local-cache>
    <invalidation-cache name="entity">
      <transaction mode="NON_XA"/>
      <object-memory size="1000"/>
      <expiration max-idle="100000"/>
    </invalidation-cache>
    <replicated-cache name="timestamps" mode="ASYNC"/>
  </cache-container>
</subsystem>
```

Note the default cache defined in each cache container. For example, the **web** cache container defines the **dist** distributed cache as the default. The **dist** cache will therefore be used when clustering web sessions.

See [Configure Cache Containers](#) for information on changing the default cache and adding additional caches.



IMPORTANT

You can add additional caches and cache containers, for example, for HTTP sessions, stateful session beans, or singleton services or deployments. It is not supported to use these caches directly by user applications.

24.3.2.1. Configure Cache Containers

Cache containers and cache attributes can be configured using the management console or management CLI.



WARNING

You should avoid changing cache or cache container names, as other components in the configuration may reference them.

Configure Caches Using the Management Console

Once you navigate to the **Infinispan** subsystem from the **Configuration** tab in the management console, you can configure caches and cache containers. In a managed domain, make sure to select the appropriate profile to configure.

- Add a cache container.
Click the Add (+) button next to the **Cache Container** heading, choose **Add Cache Container**, and enter the settings for the new cache container.
- Update cache container settings.
Choose the appropriate cache container and click **View**. Configure the cache container settings as necessary.
- Update cache container transport settings.
Choose the appropriate cache container and click **View**. Select the **Transport** tab and configure the cache container transport settings as necessary.
- Configure caches.
Choose the appropriate cache container and click **View**. From the appropriate cache tab, for example, **Replicated Cache**, you can add, update, and remove caches.

Configure Caches Using the Management CLI

You can configure caches and cache containers using the management CLI. In a managed domain, you must specify the profile to update by preceding these commands with **/profile=PROFILE_NAME**.

- Add a cache container.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:add
```

- Add a replicated cache.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/replicated-  
cache=CACHE:add(mode=MODE)
```

- Set the default cache for a cache container.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-attribute(name=default-  
cache,value=CACHE)
```

- Configure batching for a replicated cache.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/replicated-  
cache=CACHE/component=transaction:write-attribute(name=mode,value=BATCH)
```

The following example shows how to add a **concurrent** distributed cache to the **web** cache container. This cache configuration relaxes the locking constraints of the default cache, which allows multiple concurrent requests to access the same web session simultaneously. It achieves this by permitting lock-free reads and obtaining exclusive locks more frequently, but for a shorter duration.

Use the following management CLI commands to add the **concurrent** distributed cache to the **web** cache container and make it the default cache:

```
batch  
/subsystem=infinispan/cache-container=web/distributed-cache=concurrent:add  
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=concurrent)  
/subsystem=infinispan/cache-container=web/distributed-cache=concurrent/store=file:add  
run-batch
```

This results in the following server configuration:

```
<cache-container name="web" default-cache="concurrent"  
module="org.wildfly.clustering.web.infinispan">  
...  
<distributed-cache name="concurrent">  
  <file-store/>  
</distributed-cache>  
</cache-container>
```

Change the Default EJB Cache Container

You can use cache containers in the **ejb3** subsystem as described below:

- To support passivation of EJB session beans, you can use the **ejb** cache container defined in the **infinispan** subsystem to store the sessions.
- For remote EJB clients connecting to a clustered deployment on a server, you must provide the cluster topology information to these clients, so that they can fail over to other nodes in the cluster if the node they are interacting with fails.

If you are changing or renaming the default cache container, named **ejb**, which supports passivation and the provision of topology information, you must add the **cache-container** attribute to the **passivation-stores** element and the **cluster** attribute to the **remote** element, as shown in the example below. If you

are just adding a new cache for your own use, you need not make those changes.

```
<subsystem xmlns="urn:jboss:domain:ejb3:5.0">
  <passivation-stores>
    <passivation-store name="infinispan" cache-container="ejb-cltest" max-size="10000"/>
  </passivation-stores>

  <remote cluster="ejb-cltest" connector-ref="http-remoting-connector" thread-pool-name="default"/>
</subsystem>

<subsystem xmlns="urn:jboss:domain:infinispan:7.0">
  ...
  <cache-container name="ejb-cltest" aliases="sfsb" default-cache="dist"
module="org.wildfly.clustering.ejb.infinispan">
</subsystem>
```

Eviction Feature in Hibernate Cache Container

The eviction feature for the **hibernate** cache container removes cache entries from memory. This feature helps reduce the memory load on the subsystem.

The **size** attribute sets the maximum number of cache entries that will be stored before eviction of cache entries begins.

Example: Eviction Feature

```
<cache-container name="hibernate" default-cache="local-query" module="org.hibernate.infinispan">
  <transport lock-timeout="60000"/>
  <local-cache name="local-query">
    <object-memory size="1000"/>
    <expiration max-idle="100000"/>
  </local-cache>
</cache-container>
```

Note that eviction only occurs within the memory. A cache store holds cache entries that are evicted to prevent permanent loss of information. For more information about the eviction feature, see the [Eviction and Data Container](#) section in the Infinispan User Guide.

24.3.3. Clustering Modes

Clustering can be configured in two different ways in JBoss EAP using Infinispan. The best method for your application will depend on your requirements. There is a trade-off between availability, consistency, reliability and scalability with each mode. Before choosing a clustering mode, you must identify what are the most important features of your network for you, and balance those requirements.

Cache Modes

Replication

Replication mode automatically detects and adds new instances on the cluster. Changes made to these instances will be replicated to all nodes on the cluster. Replication mode typically works best in small clusters because of the amount of information that has to be replicated over the network. Infinispan can be configured to use UDP multicast, which alleviates network traffic congestion to a degree.

Distribution

Distribution mode allows Infinispan to scale the cluster linearly. Distribution mode uses a consistent hash algorithm to determine where in a cluster a new node should be placed. The number of copies, or owners, of information to be kept is configurable. There is a trade-off between the number of

copies kept, durability of the data, and performance. The more copies that are kept, the more impact on performance, but the less likely you are to lose data in a server failure. The hash algorithm also works to reduce network traffic by locating entries without multicasting or storing metadata. You should consider using distribution mode as a caching strategy when the cluster size exceeds 6-8 nodes. With distribution mode, data is distributed to only a subset of nodes within the cluster, as opposed to all nodes.

Scattered

Scattered mode is similar to distribution mode in that it uses a consistent hash algorithm to determine ownership. However, ownership is limited to two members, and the originator, or node receiving the request for a given session, always assumes ownership for coordinating locking and cache entry updates. The cache write algorithm used in scattered mode guarantees that a write operation results in only a single RPC call, where distribution caches with two owners can often use two RPC calls. This can be useful for distributed web sessions because load balancer failover tends to direct traffic to a non-primary owner or even a backup node. This can potentially reduce contention and improve performance following a cluster topology change.

Scattered mode does not support transactions or L1 caching. However, it does support biased reads, which allow the node that originates a cache write for a given entry to continue to service reads for that entry for some duration even though it is not the owner according to the consistent hash. The effect is similar to L1 caching, although the configuration attributes for biased reads versus L1 caching are distinct.

Synchronous and Asynchronous Replication

Replication can be performed either in synchronous or asynchronous mode, and the mode chosen depends on your requirements and your application.



IMPORTANT

From JBoss EAP 7.1 onwards, you must always use the synchronous (**SYNC**) cache mode. **SYNC** is also the default cache mode. Using the asynchronous (**ASYNC**) mode is not valid and causes locking contention. For more information about session replication and the appropriate cache modes, see [How to configure and tune the session replication for EAP](#).

Synchronous replication

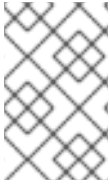
With synchronous replication, the replication process runs in the same thread that handles the user request. Session replication is initiated after the completed response is sent back to the client, and the thread is released only after the replication is completed. Synchronous replication has an impact on network traffic because it requires a response from each node in the cluster. It has the advantage, however, of ensuring that all modifications have been made to all nodes in the cluster.

Asynchronous replication

With asynchronous replication, Infinispan uses a thread pool to carry out replication in the background. The sender does not wait for replies from other nodes in the cluster. However, cache reads for the same session will block until the previous replication completes so that stale data is not read. Replication is triggered either on a time basis or by queue size. Failed replication attempts are written to a log, not notified in real time.

24.3.3.1. Configure the Cache Mode

You can change the default cache using the management CLI.



NOTE

This section shows instructions specific to configuring the web session cache, which defaults to distribution mode. The steps and management CLI commands can easily be adjusted to apply to other cache containers.

Change to Replication Cache Mode

The default JBoss EAP 7 configuration for the web session cache does not include a **repl** replication cache. This cache must first be added.



NOTE

The below management CLI commands are for a standalone server. When running in a managed domain, you must specify which profile to update by preceding the **/subsystem=infinispan** commands with **/profile=PROFILE_NAME**.

1. Add the **repl** replication cache and set it as the default cache.

```
batch
/subsystem=infinispan/cache-container=web/replicated-cache=repl:add(mode=ASYNC)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/component=transaction:add(mode=BATCH)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/component=locking:add(isolation=REPEATABLE_READ)
/subsystem=infinispan/cache-container=web/replicated-cache=repl/store=file:add
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=repl)
run-batch
```

2. Reload the server.

```
reload
```

Change to Distribution Cache Mode

The default JBoss EAP 7 configuration for the web session cache already includes a **dist** distribution cache.



NOTE

The below management CLI commands are for a standalone server. When running in a managed domain, you must specify which profile to update by preceding the **/subsystem=infinispan** commands with **/profile=PROFILE_NAME**.

1. Change the default cache to the **dist** distribution cache.

```
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=dist)
```

2. Set the number of owners for the distribution cache. The following command sets **5** owners. The default is **2**.

```
/subsystem=infinispan/cache-container=web/distributed-cache=dist:write-
attribute(name=owners,value=5)
```

3. Reload the server.

reload

Change to Scattered Cache Mode

The default JBoss EAP configuration for the web session cache does not include a **scattered-cache**. The example below shows the management CLI commands to add a scattered cache and set it as the default cache.



NOTE

The below management CLI commands are for a standalone server using the HA profile. When running in a managed domain, you must specify which profile to update by preceding the **/subsystem=infinispan** commands with **/profile=PROFILE_NAME**.

1. Create the scattered cache with a read bias lifespan equal to the default web session timeout value of 30 minutes .

```
/subsystem=infinispan/cache-container=web/scattered-cache=scattered:add(bias-lifespan=1800000)
```

2. Set **scattered** as the default cache.

```
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=scattered)
```

This results in the following server configuration.

```
<cache-container name="web" default-cache="scattered"
module="org.wildfly.clustering.web.infinispan">
...
<scattered-cache name="scattered" bias-lifespan="1800000"/>
...
</cache-container>
```

24.3.3.2. Cache Strategy Performance

When using a **SYNC** caching strategy, the cost of replication is easy to measure and directly seen in response times since the request does not complete until the replication completes.

Although it seems that the **ASYNC** caching strategy should result in lower response times than the **SYNC** caching strategy, this is only true under the right conditions. The **ASYNC** caching strategy is more difficult to measure, but it can provide better performance than the **SYNC** strategy when the duration between requests is long enough for the cache operation to complete. This is because the cost of replication is not immediately seen in response times.

If requests for the same session are made too quickly, the cost of replication for the previous request is shifted to the front of the subsequent request since it must wait for the replication from the previous request to complete. For rapidly fired requests where a subsequent request is sent immediately after a response is received, the **ASYNC** caching strategy will perform worse than the **SYNC** caching strategy. Consequently, there is a threshold for the period of time between requests for the same session where the **SYNC** caching strategy will actually perform better than the **ASYNC** caching strategy. In real world usage, requests for the same session are not normally received in rapid succession. Instead, there is typically a period of time in the order of a few seconds or more between the requests. In this case, the **ASYNC** caching strategy is a sensible default and provides the fastest response times.

24.3.4. State Transfer

State transfer is both a basic data grid and clustered cache function. Without state transfer, data would be lost as nodes are added to or removed from the cluster.

State transfer adjusts the cache's internal state in response to a change in the cache membership. This change automatically occurs when a node joins or leaves the cluster, when two or more cluster partitions merge, or after any combination of these events. The initial state transfer for a newly started cache is the most expensive, as the new cache must receive the maximum amount of state, based on the cache's mode as discussed below.

The **timeout** attribute can be used to control how long the newly started cache waits to receive its state. If the **timeout** attribute is a positive number, then the cache will wait to receive all of its initial state before being made available to service requests. If the state transfer does not complete in the specified time, the default being **240000** milliseconds, then the cache will throw an error and cancel the startup. If **timeout** is set to **0**, then the cache is immediately available, and it will receive its initial state during background operations. Until the initial state transfer is complete, any requests for cache entries that the cache has not yet received will need to be fetched from a remote node.

The **timeout** attribute can be set to **0** with the following command.

```
/subsystem=infinispan/cache-container=server/CACHE_TYPE=CACHE/component=state-  
transfer:write-attribute(name=timeout,value=0)
```

The state transfer behavior is determined by the cache's mode.

- In replication mode a new node joining the cache receives the entire cache state from the existing nodes. When a node leaves the cluster there is no state transfer.
- In distribution mode the new node receives only a part of the state from the existing nodes, and the existing nodes remove some of their state in order to keep **owners** copies of each key in the cache, as determined through consistent hashing. When a node leaves the cluster, a distribution cache needs to make additional copies of the keys that were stored on that node, so that owners of each key continue to exist.
- In invalidation mode the initial state transfer is similar to replication mode, the only difference being that the nodes are not guaranteed to have the same state. When a node leaves the cluster there is no state transfer.

A state transfer transfers both in-memory and persistent state by default, but both can be disabled in the configuration. When state transfer is disabled, a **ClusterLoader** must be configured, otherwise a node will become the owner or backup owner of a key without the data being loaded into its cache. In addition, if state transfer is disabled in distribution mode then a key will occasionally have less than **owners** copies in the cache.

24.3.5. Configure Infinispan Thread Pools

The **infinispan** subsystem contains the **async-operations**, **expiration**, **listener**, **persistence**, **remote-command**, **state-transfer**, and **transport** thread pools. These pools can be configured for any Infinispan cache container.

The following table lists the attributes you can configure for each thread pool in the **infinispan** subsystem and the default value for each.

Thread Pool Name	keepalive-time	max-threads	min-threads	queue-length
async-operations	60000L	25	25	1000
expiration	60000L	1	N/A	N/A
listener	60000L	1	1	100000
persistence	60000L	4	1	0
remote-command	60000L	200	1	0
state-transfer	60000L	60	1	0
transport	60000L	25	25	100000

Use the following syntax to configure an Infinispan thread pool using the management CLI.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER_NAME/thread-
pool=THREAD_POOL_NAME:write-attribute(name=ATTRIBUTE_NAME,
value=ATTRIBUTE_VALUE)
```

The following is an example of the management CLI command to set the **max-threads** value to **10** in the **persistence** thread pool for the **server** cache container.

```
/subsystem=infinispan/cache-container=server/thread-pool=persistence:write-attribute(name="max-
threads", value="10")
```

24.3.6. Infinispan Statistics

Runtime statistics about Infinispan caches and cache containers can be enabled for monitoring purposes. Statistics collection is not enabled by default for performance reasons.

Statistics collection can be enabled for each cache container, cache, or both. The statistics option for each cache overrides the option for the cache container. Enabling or disabling statistics collection for a cache container will cause all caches in that container to inherit the setting, unless they explicitly specify their own.

24.3.6.1. Enable Infinispan Statistics



WARNING

Enabling Infinispan statistics may have a negative impact on the performance of the **infinispan** subsystem. Statistics should be enabled only when required.

You can enable or disable the collection of Infinispan statistics using the management console or the management CLI. From the management console, navigate to the **Infinispan** subsystem from the **Configuration** tab, select the appropriate cache or cache container, and edit the **Statistics Enabled** attribute. Use the below commands to enable statistics using the management CLI.

Enable statistics collection for a cache container. A server reload will be required.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-attribute(name=statistics-enabled,value=true)
```

Enable statistics collection for a cache. A server reload will be required.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/CACHE_TYPE=CACHE:write-attribute(name=statistics-enabled,value=true)
```

NOTE

You can use the following command to undefine the **statistics-enabled** attribute of a cache so that it will inherit the settings of its cache container's **statistics-enabled** attribute.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/CACHE_TYPE=CACHE:undefine-attribute(name=statistics-enabled)
```

24.3.7. Infinispan Partition Handling

An *Infinispan cluster* is built out of several nodes where data is stored. To prevent data loss if multiple nodes fail, Infinispan copies the same data over multiple nodes. This level of data redundancy is configured using the **owners** attribute. As long as fewer than the configured number of nodes crash simultaneously, Infinispan will have a copy of the data available.

However, there are potential catastrophic situations that could occur when too many nodes disappear from the cluster:

Split brain

This splits the cluster in two or more partitions, or sub-clusters, that operate independently. In these circumstances, multiple clients reading and writing from different partitions see different versions of the same cache entry, which for many applications is problematic.

NOTE

There are ways to alleviate the possibility for the split brain to happen, such as redundant networks or [IP bonding](#); however, these only reduce the window of time for the problem to occur.

Multiple nodes crash in sequence

If multiple nodes, specifically the number of owners, crash in rapid sequence and Infinispan does not have the time to properly rebalance its state between crashes, the result is partial data loss.

The goal is to avoid situations in which incorrect data is returned to the user as a result of either split brain or multiple nodes crashing in rapid sequence.

24.3.7.1. Split Brain

In a split brain situation, each network partition will install its own JGroups view, removing the nodes from the other partitions. We do not have a direct way to determine whether the cluster has been split into two or more partitions, since the partitions are unaware of each other. Instead, we assume the cluster has split when one or more nodes disappear from the JGroups cluster without sending an explicit leave message.

With partition handling disabled, each such partition would continue to function as an independent cluster. Each partition may only see a part of the data, and each partition could write conflicting updates in the cache.

With partition handling enabled, if we detect a split, each partition does not start a rebalance immediately, but first checks whether it should enter degraded mode instead:

- If at least one segment has lost all its owners, meaning that at least the number of owners specified has left since the last rebalance ended, the partition enters degraded mode.
- If the partition does not contain a simple majority of the nodes ($\text{floor}(\text{numNodes}/2) + 1$) in the *latest stable topology*, the partition also enters degraded mode.
- Otherwise, the partition keeps functioning normally and starts a rebalance.

The *stable topology* is updated every time a rebalance operation ends and the coordinator determines that another rebalance is not necessary. These rules ensure that at most, one partition stays in available mode, and the other partitions enter degraded mode.

When a partition is in degraded mode, it only allows access to the keys that are wholly owned:

- Requests (reads and writes) for entries that have all the copies on nodes within this partition are honored.
- Requests for entries that are partially or totally owned by nodes that have disappeared are rejected with an **AvailabilityException**.

This guarantees that partitions cannot write different values for the same key (cache is consistent), and also that one partition can not read keys that have been updated in the other partitions (no stale data).



NOTE

Two partitions could start up isolated, and as long as they do not merge, they can read and write inconsistent data. In the future, we may allow custom availability strategies (e.g. check that a certain node is part of the cluster, or check that an external machine is accessible) that could handle that situation as well.

24.3.7.2. Configuring Partition Handling

Currently the partition handling is disabled by default. Use the following management CLI command to enable partition handling:

```
/subsystem=infinispan/cache-container=web/distributed-cache=dist/component=partition-handling:write-attribute(name=enabled, value=true)
```

24.3.8. Configuring Remote Cache Containers

You must configure a remote cache for every server group in a managed domain. You can enable a collection of metrics for a given **remote-cache-container** and associated runtime caches with the **statistics-enabled** attribute.

24.3.8.1. Creating a Remote Cache Container

Each server group in a managed domain requires a unique remote cache. The caches can belong to the same data grid. Therefore, users must configure a remote cache for every server group by defining a socket binding for the server group and associating the socket binding with a remote cache container.

Procedure

1. Define a **socket-binding**, repeating the command as necessary for each remote Red Hat Data Grid instance in the cluster.

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=SOCKET_BINDING:add(host=HOSTNAME,port=PORT)
```

2. Define a **remote-cache-container** that references the newly created socket bindings.

```
batch
/subsystem=infinispan/remote-cache-container=CACHE_CONTAINER:add(default-remote-cluster=data-grid-cluster)
/subsystem=infinispan/remote-cache-container=CACHE_CONTAINER/remote-cluster=data-grid-cluster:add(socket-bindings=[SOCKET_BINDING,SOCKET_BINDING_2,...])
run-batch
```

24.3.8.2. Enabling Statistics for a Remote Cache Container

The **statistics-enabled** attribute enables a collection of metrics for a given **remote-cache-container** and associated runtime caches.

- For a **remote-cache-container** called "foo", enable statistics using the following operation:

```
/subsystem=infinispan/remote-cache-container=foo:write-attribute(name=statistics-enabled,value=true)
```

- For a **remote-cache-container** "foo", the following metrics are visible at runtime:

```
/subsystem=infinispan/remote-cache-container=foo:read-attribute(name=connections)
/subsystem=infinispan/remote-cache-container=foo:read-attribute(name=active-connections)
/subsystem=infinispan/remote-cache-container=foo:read-attribute(name=idle-connections)
```

- For descriptions of these metrics, execute a read-resource-description operation for the **remote-cache-container**:

```
/subsystem=infinispan/remote-cache-container=foo:read-resource-description
```

- The following metrics are specific to the remote cache used by the selected deployment:

```
/subsystem=infinispan/remote-cache-container=foo/remote-cache=bar.war:read-resource(include-runtime=true, recursive=true)
{
```

```

    "average-read-time" : 1,
    "average-remove-time" : 0,
    "average-write-time" : 2,
    "hits" : 9,
    "misses" : 0,
    "near-cache-hits" : 7,
    "near-cache-invalidations" : 8,
    "near-cache-misses" : 9,
    "near-cache-size" : 1,
    "removes" : 0,
    "time-since-reset" : 82344,
    "writes" : 8
  }

```

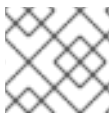
- For descriptions of these metrics, execute a read-resource-description operation for the remote cache:

```
/subsystem=infinispan/remote-cache-container=foo/remote-cache=bar.war:read-resource-description
```

- Some of these metrics are computed values (example, average-*), while others are tallied, such as hits and misses. The tallied metrics can be reset by the following operation:

```
/subsystem=infinispan/remote-cache-container=foo/remote-cache=bar.war:reset-statistics()
```

24.3.9. Externalize HTTP Sessions to Red Hat Data Grid



NOTE

You need a Red Hat Data Grid subscription to use this functionality.

Red Hat Data Grid can be used as an external cache container for application-specific data in JBoss EAP, such as HTTP sessions. This allows scaling of the data layer independent of the application, and enables different JBoss EAP clusters, which may reside in various domains, to access data from the same Red Hat Data Grid cluster. Additionally, other applications can interface with the caches presented by Red Hat Data Grid.

The following example shows how to externalize HTTP sessions. It applies to both standalone instances of JBoss EAP as well as managed domains.

1. Create a **remote-cache-container**. For more information, see [Configuring Remote Cache Containers](#).
2. Configure the HotRod store. The HotRod store uses one dedicated remote cache for each cache created by the JBoss EAP server. Typically, one invalidation cache is used on the JBoss EAP server, as shown in the CLI script below.



NOTE

The remote caches need to be configured manually on the Red Hat Data Grid servers. The recommended configuration for these caches is a transactional distribution mode cache with pessimistic locking. Cache names must correspond to the deployment file names, such as **test.war**.

Once the remote cache container has been configured, a **hotrod** store can be configured to replace any existing store. The following CLI script demonstrates a typical use case for offloading sessions in conjunction with the invalidation cache.

```
batch
/subsystem=infinispan/cache-container=web/invalidation-cache=CACHE_NAME:add()
/subsystem=infinispan/cache-container=web/invalidation-
cache=CACHE_NAME/store=hotrod:add(remote-cache-
container=CACHE_CONTAINER,fetch-
state=false,purge=false,passivation=false,shared=true)
/subsystem=infinispan/cache-container=web/invalidation-
cache=CACHE_NAME/component=transaction:add(mode=BATCH)
/subsystem=infinispan/cache-container=web/invalidation-
cache=CACHE_NAME/component=locking:add(isolation=REPEATABLE_READ)
/subsystem=infinispan/cache-container=web:write-attribute(name=default-
cache,value=CACHE_NAME)
run-batch
```

The script configures a new invalidation cache. Session data is then maintained in the cache for performance and written to the store for resilience.

A HotRod client can be injected directly into Jakarta EE applications using the **@Resource** annotation. In the example below, the **@Resource** annotation looks up the configuration properties in the class path, in the **hotrod-client.properties** file.

```
@Resource(lookup = "java:jboss/infinispan/remote-container/web-sessions")
private org.infinispan.client.hotrod.RemoteCacheContainer client;
```

Example: hotrod-client.properties File

```
infinispan.client.hotrod.transport_factory =
org.infinispan.client.hotrod.impl.transport.tcp.TcpTransportFactory
infinispan.client.hotrod.server_list = 127.0.0.1:11222
infinispan.client.hotrod.marshaller =
org.infinispan.commons.marshall.jboss.GenericJBossMarshaller
infinispan.client.hotrod.async_executor_factory =
org.infinispan.client.hotrod.impl.async.DefaultAsyncExecutorFactory
infinispan.client.hotrod.default_executor_factory.pool_size = 1
infinispan.client.hotrod.default_executor_factory.queue_size = 10000
infinispan.client.hotrod.hash_function_impl.1 =
org.infinispan.client.hotrod.impl.consistenthash.ConsistentHashV1
infinispan.client.hotrod.tcp_no_delay = true
infinispan.client.hotrod.ping_on_startup = true
infinispan.client.hotrod.request_balancing_strategy =
org.infinispan.client.hotrod.impl.transport.tcp.RoundRobinBalancingStrategy
infinispan.client.hotrod.key_size_estimate = 64
infinispan.client.hotrod.value_size_estimate = 512
infinispan.client.hotrod.force_return_values = false

## below is connection pooling config

maxActive=-1
maxTotal = -1
maxIdle = -1
whenExhaustedAction = 1
```

```
timeBetweenEvictionRunsMillis=120000
minEvictableIdleTimeMillis=300000
testWhileIdle = true
minIdle = 1
```

Securing a Remote Cache Container

It is possible to secure the communication to the remote Red Hat Data Grid instance using SSL. This is accomplished by configuring the **remote-cache-container** on the JBoss EAP instance and adjusting the hotrod connector on the Red Hat Data Grid instance to use an active security realm.

1. Create a **client-ssl-context** in JBoss EAP. For additional information on creating a **client-ssl-context**, including generating the other elytron components, see [Using a client-ssl-context](#) in *How to Configure Server Security* for JBoss EAP.

```
/subsystem=elytron/client-ssl-context=CLIENT_SSL_CONTEXT:add(key-
manager=KEY_MANAGER,trust-manager=TRUST_MANAGER)
```

2. Configure the remote cache container to use the client SSL context.

```
/subsystem=infinispan/remote-cache-
container=CACHE_CONTAINER/component=security:write-attribute(name=ssl-
context,value=CLIENT_SSL_CONTEXT)
```

3. Secure the remote Red Hat Data Grid instance, repeating as necessary for each instance.
 - a. Copy the keystore used in the **client-ssl-context** to the remote Red Hat Data Grid instance.
 - b. Configure the **ApplicationRealm** to use this keystore.

```
/core-service=management/security-realm=ApplicationRealm/server-
identity=ssl:add(keystore-path="KEYSTORE_NAME",keystore-relative-
to="jboss.server.config.dir",keystore-password="KEYSTORE_PASSWORD")
```

- c. Adjust the hotrod connector to point to this security realm.

```
/subsystem=datagrid-infinispan-endpoint/hotrod-connector=hotrod-
connector/encryption=ENCRYPTION:add(require-ssl-client-auth=false,security-
realm="ApplicationRealm")
```

- d. Reload the remote Red Hat Data Grid instance.

```
reload
```

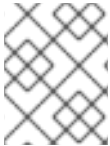
24.3.10. Externalize HTTP Sessions to Red Hat Data Grid Using a Remote Store



NOTE

You need a Red Hat Data Grid subscription to use this functionality.

The instructions here represent an older way of externalizing sessions. JBoss EAP 7.2 introduced a custom optimized cache store based on the HotRod protocol that integrates with the **elytron** subsystem. It is recommended to use the new **hotrod** store, as described in [Externalize HTTP Sessions to Red Hat Data Grid](#).

**NOTE**

For each distributable application, an entirely new cache must be created. It can be created in an existing cache container, for example, **web**.

To externalize HTTP sessions:

1. Define the location of the remote Red Hat Data Grid server by adding the networking information to the **socket-binding-group**.

Example: Adding Remote Socket Bindings

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-rhdg-server1:add(host=RHDGHostName1, port=11222)
```

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-rhdg-server2:add(host=RHDGHostName2, port=11222)
```

Resulting XML

```
<socket-binding-group name="standard-sockets" ... >
  ...
  <outbound-socket-binding name="remote-rhdg-server1">
    <remote-destination host="RHDGHostName1" port="11222"/>
  </outbound-socket-binding>
  <outbound-socket-binding name="remote-rhdg-server2">
    <remote-destination host="RHDGHostName2" port="11222"/>
  </outbound-socket-binding>
</socket-binding-group>
```

**NOTE**

You will need a remote socket binding configured for each Red Hat Data Grid server.

2. Ensure the remote cache containers are defined in JBoss EAP's **infinispan** subsystem; in the example below the **cache** attribute in the **remote-store** element defines the cache name on the remote Red Hat Data Grid server.

If you are running in a managed domain, precede these commands with **/profile=PROFILE_NAME**.

Example: Adding a Remote Cache Container

```
/subsystem=infinispan/cache-container=web/invalidation-cache=rhdg:add(mode=SYNC)
```

```
/subsystem=infinispan/cache-container=web/invalidation-cache=rhdg/component=locking:write-attribute(name=isolation,value=REPEATABLE_READ)
```

```
/subsystem=infinispan/cache-container=web/invalidation-cache=rhdg/component=transaction:write-attribute(name=mode,value=BATCH)
```

```
/subsystem=infinispan/cache-container=web/invalidation-
```

```
cache=rhdg/store=remote:add(remote-servers=["remote-rhdg-server1","remote-rhdg-
server2"], cache=default, socket-timeout=60000, passivation=false, purge=false,
shared=true)
```

Resulting XML

```
<subsystem xmlns="urn:jboss:domain:infinispan:7.0">
  ...
  <cache-container name="web" default-cache="dist"
module="org.wildfly.clustering.web.infinispan" statistics-enabled="true">
    <transport lock-timeout="60000"/>
    <invalidation-cache name="rhdg" mode="SYNC">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <remote-store cache="default" socket-timeout="60000" remote-servers="remote-rhdg-
server1 remote-rhdg-server2" passivation="false" purge="false" shared="true"/>
    </invalidation-cache>
    ...
  </cache-container>
</subsystem>
```

3. Add cache information into the application **jboss-web.xml** file. In the following example, **web** is the name of the cache container and **rhdg** is the name of the appropriate cache located in this container.

Example: jboss-web.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web xmlns="http://www.jboss.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-web_10_0.xsd"
  version="10.0">
  <replication-config>
    <replication-granularity>SESSION</replication-granularity>
    <cache-name>web.rhdg</cache-name>
  </replication-config>
</jboss-web>
```

24.4. CONFIGURING JBOSS EAP AS A FRONT-END LOAD BALANCER

You can configure JBoss EAP and the **undertow** subsystem to act as a front-end load balancer to proxy requests to back-end JBoss EAP servers. Since Undertow makes use of asynchronous IO, the IO thread that is responsible for the connection is the only thread that is involved in the request. That same thread is also used for the connection made to the back-end server.

You can use the following protocols:

- HTTP over plain text (**http**), supporting HTTP/1 and HTTP/2 (**h2c**)
- HTTP over secured connection (**https**), supporting HTTP/1 and HTTP/2 (**h2**)
- AJP (**ajp**)

You can either define a [static load balancer](#) and specify the back-end hosts in your configuration, or use the [mod_cluster front end](#) to dynamically update the hosts.

See [Configuring HTTP/2](#) for instructions to configure Undertow to use HTTP/2.

24.4.1. Configure Undertow as a Load Balancer Using mod_cluster

You can use the built-in mod_cluster front-end load balancer to load balance other JBoss EAP instances.

This procedure assumes that you are running in a managed domain and already have the following configured:

- A JBoss EAP server that will act as the load balancer.
 - This server uses the **load-balancer** profile, which is bound to the **load-balancer-sockets** socket binding group.



NOTE

The **load-balancer** profile is already preconfigured with the socket binding, mod-cluster Undertow filter, and reference in the default host to the filter in order to use this server as a front-end load balancer.

- Two JBoss EAP servers, which will act as the back-end servers.
 - These servers are running in a cluster and use the **ha** profile, which is bound to the **ha-sockets** socket binding group.
- The distributable application to be load balanced deployed to the back-end servers.

Configure the mod_cluster Front-end Load Balancer

The below steps load balance servers in a managed domain, but they can be adjusted to apply to a set of standalone servers. Be sure to update the management CLI command values to suit your environment.

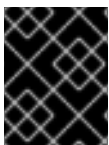
1. Set the mod_cluster advertise security key.
Adding the advertise security key allows the load balancer and servers to authenticate during discovery.

Use the following management CLI command to set the mod_cluster advertise security key.

```
/profile=ha/subsystem=modcluster/proxy=default:write-attribute(name=advertise-security-key,
value=mypassword)
```

2. Update the mod_cluster load balancer's security key.
Use the following management CLI command to set the security key for the mod_cluster filter.

```
/profile=load-balancer/subsystem=undertow/configuration=filter/mod-cluster=load-
balancer:write-attribute(name=security-key,value=mypassword)
```



IMPORTANT

It is recommended that the management and advertise socket bindings used by mod_cluster only be exposed to the internal network, not a public IP address.

The load balancer JBoss EAP server can now load balance the two back-end JBoss EAP servers.

Multiple `mod_cluster` Configurations

The **`mod_cluster`** subsystem supports multiple named proxy configurations which allows to register non-default **`undertow`** servers with the reverse proxies. Moreover, this allows single application server node to register with different groups of proxy servers.

The following example adds an **`ajp-listener`**, a server, and a host to the **`undertow`** server. It also adds a new **`mod_cluster`** configuration that registers the host using the advertise mechanism.

```
/socket-binding-group=standard-sockets/socket-binding=ajp-other:add(port=8010)
/subsystem=undertow/server=other-server:add
/subsystem=undertow/server=other-server/ajp-listener=ajp-other:add(socket-binding=ajp-other)
/subsystem=undertow/server=other-server/host=other-host:add(default-web-module=root-other.war)
/subsystem=undertow/server=other-server/host=other-host
/location=other:add(handler=welcome-content)
/subsystem=undertow/server=other-server/host=other-host:write-attribute(name=alias,value=
[localhost])

/socket-binding-group=standard-sockets/socket-binding=modcluster-other:add(multicast-
address=224.0.1.106,multicast-port=23364)
/subsystem=modcluster/proxy=other:add(advertise-socket=modcluster-other,balancer=other-
balancer,connector=ajp-other)

reload
```

24.4.2. Enabling Ranked Session Affinity in Your Load Balancer

You must enable ranked session affinity in your load balancer to have session affinity with multiple, ordered routes in the **`distributable-web`** subsystem. For more information about the **`distributable-web`** subsystem and the different affinity options, see [The distributable-web subsystem for Distributable Web Session Configurations](#) in the *Development Guide* for JBoss EAP.

The default delimiter that separates the node routes is `..`. If you want a different value, you can configure the **`delimiter`** attribute of the **`affinity`** resource.

Procedure

1. Enable ranked session affinity for a load balancer:

```
/subsystem=undertow/configuration=filter/mod-cluster=load-balancer/affinity=ranked:add
```

2. Optional: Configure the **`delimiter`** attribute of the **`affinity`** resource:

```
/subsystem=undertow/configuration=filter/mod-cluster=load-balancer/affinity=ranked:write-
attribute(name=delimiter,value=':')
```

24.4.3. Configure Undertow as a Static Load Balancer

To configure a static load balancer with Undertow, you need to configure a proxy handler in the **`undertow`** subsystem. To configure a proxy handler in Undertow, you need to do the following on your JBoss EAP instance that will serve as your static load balancer:

1. Add a reverse proxy handler

2. Define the outbound socket bindings for each remote host
3. Add each remote host to the reverse proxy handler
4. Add the reverse proxy location

The following example shows how to configure a JBoss EAP instance to be a static load balancer. The JBoss EAP instance is located at **lb.example.com** and will load balance between two additional servers: **server1.example.com** and **server2.example.com**. The load balancer will reverse-proxy to the location **/app** and will use the AJP protocol.

1. To add a reverse proxy handler:

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-handler:add
```

2. To define the outbound socket bindings for each remote host:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-host1:add(host=server1.example.com, port=8009)
```

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-host2:add(host=server2.example.com, port=8009)
```

3. To add each remote host to the reverse proxy handler:

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-handler/host=host1:add(outbound-socket-binding=remote-host1, scheme=ajp, instance-id=myroute1, path=/test)
```

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-handler/host=host2:add(outbound-socket-binding=remote-host2, scheme=ajp, instance-id=myroute2, path=/test)
```

4. To add the reverse proxy location:

```
/subsystem=undertow/server=default-server/host=default-host/location=/test:add(handler=my-handler)
```

When accessing **lb.example.com:8080/app**, you will now see the content proxied from **server1.example.com** and **server2.example.com**.

24.5. USING AN EXTERNAL WEB SERVER AS A PROXY SERVER

JBoss EAP can accept requests from an external web server using the supported HTTP, HTTPS, or AJP protocol, depending on the external web server configuration.

See [Overview of HTTP Connectors](#) for details on the supported HTTP connectors for each web server. Once you have decided which web server and HTTP connector to use, see the appropriate section for information on configuring your connector:

- See the [mod_cluster](#), [mod_jk](#), or [mod_proxy](#) section for [Apache HTTP Server](#).
- See the [ISAPI connector](#) section for Microsoft IIS.

- See the [NSAPI connector](#) section for Oracle iPlanet Web Server.

For the most current information about supported configurations for HTTP connectors, see [JBoss EAP supported configurations](#).

You also will need to make sure that JBoss EAP is configured to [accept requests from external web servers](#).

24.5.1. Overview of HTTP Connectors

JBoss EAP has the ability to use load-balancing and clustering mechanisms built into external web servers, such as Apache HTTP Server, Microsoft IIS, and Oracle iPlanet as well as through Undertow. JBoss EAP communicates with the web servers using a connector. These connectors are configured within the **undertow** subsystem of JBoss EAP.

The web servers include software modules which control the way that HTTP requests are routed to JBoss EAP nodes. Each of these modules varies in how it works and how it is configured. The modules are configured to balance work loads across multiple JBoss EAP nodes, to move work loads to alternate servers in case of a failure event, or both.

JBoss EAP supports several different connectors. The one you choose depends on the web server in use and the functionality you need. See the tables below for comparisons of the supported configurations and features of the various HTTP connectors that are compatible with JBoss EAP.



NOTE

See [Configure Undertow as a Load Balancer Using mod_cluster](#) for using JBoss EAP 7 as a multi-platform load balancer.

For the most current information about supported configurations for HTTP connectors, see [JBoss EAP supported configurations](#).

Table 24.1. HTTP Connector Supported Configurations

Connector	Web Server	Supported Operating Systems	Supported Protocols
mod_cluster	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server, JBoss EAP (Undertow)	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	HTTP, HTTPS, AJP, WebSocket
mod_jk	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	AJP
mod_proxy	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	HTTP, HTTPS, AJP

Connector	Web Server	Supported Operating Systems	Supported Protocols
ISAPI connector	Microsoft IIS	Microsoft Windows Server	AJP
NSAPI connector	Oracle iPlanet Web Server	Oracle Solaris	AJP

Table 24.2. HTTP Connector Features

Connector	Supports Sticky Sessions	Adapts to Deployment Status
mod_cluster	Yes	Yes. Detects deployment and undeployment of applications and dynamically decides whether to direct client requests to a server based on whether the application is deployed on that server.
mod_jk	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.
mod_proxy	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.
ISAPI connector	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.
NSAPI connector	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.

24.5.2. Apache HTTP Server

A standalone Apache HTTP Server bundle is now available as a separate download with Red Hat JBoss Core Services. This simplifies installation and configuration, and allows for a more consistent update experience.

24.5.2.1. Installing Apache HTTP Server

For information on installing Apache HTTP Server, see the JBoss Core Services [Apache HTTP Server Installation Guide](#).

24.5.3. Accepting Requests from External Web Servers

JBoss EAP does not require any special configuration to begin accepting requests from a proxy server as long as the correct protocol handler, for example AJP, HTTP, or HTTPS, is configured.

If the proxy server uses `mod_jk`, `mod_proxy`, `ISAPI`, or `NSAPI`, it sends requests to JBoss EAP and JBoss EAP simply provides a response. With `mod_cluster`, you must also configure your network to allow JBoss EAP to send information, such as its current load, application lifecycle events, and health status, to the proxy server to help it determine where to route requests. For more information about configuring a `mod_cluster` proxy server, see [The `mod_cluster` HTTP Connector](#).

Update JBoss EAP Configuration

In the following procedure, substitute the protocols and ports in the examples with the ones you need to configure.

1. Configure the **instance-id** attribute of Undertow.

The external web server identifies the JBoss EAP instance in its connector configuration using the **instance-id**. Use the following management CLI command to set the **instance-id** attribute in Undertow.

```
/subsystem=undertow:write-attribute(name=instance-id,value=node1)
```

In the above example, the external web server identifies the current JBoss EAP instance as **node1**.

2. Add the necessary listeners to Undertow.

In order for an external web server to be able to connect to JBoss EAP, Undertow needs a listener. Each protocol needs its own listener, which is tied to a socket binding.



NOTE

Depending on your desired protocol and port configuration, this step may not be necessary. An HTTP listener is configured in all default JBoss EAP configurations, and an AJP listener is configured if you are using the *ha* or *full-ha* profile.

You can check whether the required listeners are already configured by reading the default server configuration:

```
/subsystem=undertow/server=default-server:read-resource
```

To add a listener to Undertow, it must have a socket binding. The socket binding is added to the socket binding group used by your server or server group. The following management CLI command adds an **ajp** socket binding, bound to port **8009**, to the **standard-sockets** socket binding group

```
/socket-binding-group=standard-sockets/socket-binding=ajp:add(port=8009)
```

The following management CLI command adds an **ajp** listener to Undertow, using the **ajp** socket binding.

```
/subsystem=undertow/server=default-server/ajp-listener=ajp:add(socket-binding=ajp)
```

24.6. THE MOD_CLUSTER HTTP CONNECTOR

The `mod_cluster` connector is an Apache HTTP Server-based load balancer. It uses a communication channel to forward requests from the Apache HTTP Server to one of a set of application server nodes.

The `mod_cluster` connector has several advantages over other connectors.

- The `mod_cluster` Management Protocol (MCMP) is an additional connection between the JBoss EAP servers and the Apache HTTP Server with the `mod_cluster` module enabled. It is used by the JBoss EAP servers to transmit server-side load-balance factors and lifecycle events back to the Apache HTTP Server via a custom set of HTTP methods.

- Dynamic configuration of Apache HTTP Server with `mod_cluster` allows JBoss EAP servers to join the load-balancing arrangement without manual configuration.
- JBoss EAP performs the load-balancing factor calculations, rather than relying on the Apache HTTP Server with `mod_cluster`. This makes load-balancing metrics more accurate than other connectors.
- The `mod_cluster` connector gives fine-grained application lifecycle control. Each JBoss EAP server forwards web application context lifecycle events to the Apache HTTP Server, informing it to start or stop routing requests for a given context. This prevents end users from seeing HTTP errors due to unavailable resources.
- AJP, HTTP or HTTPS transports can be used.

For more details on the specific configuration options of the **modcluster** subsystem, see the [ModCluster Subsystem Attributes](#).

24.6.1. Configure `mod_cluster` in Apache HTTP Server

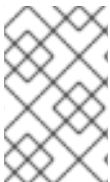
The `mod_cluster` modules are already included when installing JBoss Core Services Apache HTTP Server or using JBoss Web Server and are loaded by default.



NOTE

Apache HTTP Server is no longer distributed with JBoss Web Server as of version 3.1.0.

See the steps below to configure the `mod_cluster` module to suit your environment.



NOTE

Red Hat customers can also use the [Load Balancer Configuration Tool](#) on the Red Hat Customer Portal to quickly generate optimal configuration templates for `mod_cluster` and other connectors. Note that you must be logged in to access this tool.

Configure `mod_cluster`

Apache HTTP Server already contains a `mod_cluster` configuration file, **`mod_cluster.conf`**, that loads the `mod_cluster` modules and provides basic configuration. The IP address, port, and other settings in this file, shown below, can be configured to suit your needs.

```
# mod_proxy_balancer should be disabled when mod_cluster is used
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule cluster_slotmem_module modules/mod_cluster_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule advertise_module modules/mod_advertise.so

MemManagerFile cache/mod_cluster

<IfModule manager_module>
Listen 6666
<VirtualHost *:6666>
  <Directory />
    Require ip 127.0.0.1
  </Directory>
  ServerAdvertise on
```

```

EnableMCPMReceive
<Location /mod_cluster_manager>
  SetHandler mod_cluster-manager
  Require ip 127.0.0.1
</Location>
</VirtualHost>
</IfModule>

```

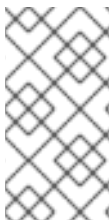
The Apache HTTP Server server is configured as a load balancer and can work with the **modcluster** subsystem running on JBoss EAP. You must [configure a mod_cluster worker node](#) to make JBoss EAP aware of mod_cluster.

If you want to disable advertising for mod_cluster and configure a static proxy list instead, see [Disable Advertisement for mod_cluster](#). For more information on the available mod_cluster configuration options in Apache HTTP Server, see the [Apache HTTP Server mod_cluster Directives](#).

For more details on configuring mod_cluster, see the [Configure Load Balancing Using Apache HTTP Server and mod_cluster](#) section of the JBoss Web Server *HTTP Connectors and Load Balancing Guide*.

24.6.2. Disable Advertising for mod_cluster

By default, the **modcluster** subsystem's balancer uses multicast UDP to advertise its availability to the background workers. You can disable advertising and to use a proxy list instead using the following procedure.



NOTE

The management CLI commands in the following procedure assume that you are using the **full-ha** profile in a managed domain. If you are using a profile other than **full-ha**, use the appropriate profile name in the command. If you are running a standalone server, remove the **/profile=full-ha** completely.

1. Modify the Apache HTTP Server configuration.

Edit the **httpd.conf** Apache HTTP Server configuration file. Make the following updates to the virtual host that listens for MCPM requests, using the **EnableMCPMReceive** directive.

- a. Add the directive to disable server advertisement.
Set the **ServerAdvertise** directive to **Off** to disable server advertisement.

```
ServerAdvertise Off
```

- b. Disable the advertise frequency.
If your configuration specifies the **AdvertiseFrequency** parameter, comment it out using a **#** character.

```
# AdvertiseFrequency 5
```

- c. Enable the ability to receive MCPM messages.
Ensure that the **EnableMCPMReceive** directive exists, to allow the web server to receive MCPM messages from the worker nodes.

```
EnableMCPMReceive
```

2. Disable advertising in the JBoss EAP **modcluster** subsystem.
Use the following management CLI command to disable advertising.

```
/profile=full-ha/subsystem=modcluster/proxy=default:write-attribute(name=advertise,value=false)
```



IMPORTANT

Be sure to continue to the next step to provide the list of proxies. Advertising will not be disabled if the list of proxies is empty.

3. Provide a list of proxies in the JBoss EAP **modcluster** subsystem.
It is necessary to provide a list of proxies because the **modcluster** subsystem will not be able to automatically discover proxies if advertising is disabled.

First, define the outbound socket bindings in the appropriate socket binding group.

```
/socket-binding-group=full-ha-sockets/remote-destination-outbound-socket-binding=proxy1:add(host=10.33.144.3,port=6666)
/socket-binding-group=full-ha-sockets/remote-destination-outbound-socket-binding=proxy2:add(host=10.33.144.1,port=6666)
```

Next, add the proxies to the `mod_cluster` configuration.

```
/profile=full-ha/subsystem=modcluster/proxy=default:list-add(name=proxies,value=proxy1)
/profile=full-ha/subsystem=modcluster/proxy=default:list-add(name=proxies,value=proxy2)
```

The Apache HTTP Server balancer no longer advertises its presence to worker nodes and UDP multicast is no longer used.

24.6.3. Configure a `mod_cluster` Worker Node

A `mod_cluster` worker node consists of a JBoss EAP server. This server can be a standalone server or part of a server group in a managed domain. A separate process runs within JBoss EAP, which manages all of the worker nodes of the cluster. This is called the master.

Worker nodes in a managed domain share an identical configuration across a server group. Worker nodes running as standalone servers are configured individually. The configuration steps are otherwise identical.

- A standalone server must be started with the *standalone-ha* or *standalone-full-ha* profile.
- A server group in a managed domain must use the *ha* or *full-ha* profile, and the *ha-sockets* or *full-ha-sockets* socket binding group. JBoss EAP ships with a cluster-enabled server group called *other-server-group* which meets these requirements.

Configure a Worker Node

The management CLI commands in this procedure assume that you are using a managed domain with the **full-ha** profile. If you are running a standalone server, remove the **/profile=full-ha** portion of the commands.

1. Configure the network interfaces.

By default, the network interfaces all default to **127.0.0.1**. Every physical host that hosts either a standalone server or one or more servers in a server group needs its interfaces to be configured to use its public IP address, which the other servers can see.

Use the following management CLI commands to modify the external IP addresses for the **management**, **public**, and **unsecure** interfaces as appropriate for your environment. Be sure to replace **EXTERNAL_IP_ADDRESS** in the command with the actual external IP address of the host.

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:EXTERNAL_IP_ADDRESS}")
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address.public:EXTERNAL_IP_ADDRESS}")
/interface=unsecure:write-attribute(name=inet-
address,value="${jboss.bind.address.unsecure:EXTERNAL_IP_ADDRESS}")
```

Reload the server.

```
reload
```

2. Configure host names.

Set a unique host name for each host that participates in a managed domain. This name must be unique across slaves and will be used for the slave to identify to the cluster, so make a note of the name you use.

- a. Start the JBoss EAP slave host, using the appropriate **host.xml** configuration file.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

- b. Use the following management CLI command to set a unique host name. This example uses **slave1** as the new host name.

```
/host=EXISTING_HOST_NAME:write-attribute(name=name,value=slave1)
```

For more information on configuring a host name, see [Configure the Name of a Host](#).

3. Configure each host to connect to the domain controller.



NOTE

This step does not apply for a standalone server.

For newly configured hosts that need to join a managed domain, you must remove the local element and add the remote element host attribute that points to the domain controller.

- a. Start the JBoss EAP slave host, using the appropriate **host.xml** configuration file.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

- b. Use the following management CLI command to configure the domain controller settings.

```
/host=SLAVE_HOST_NAME:write-remote-domain-
controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.domain.master.por
t:9990},security-realm="ManagementRealm")
```

This modifies the XML in the `host-slave.xml` file as follows:

```
<domain-controller>
  <remote host="DOMAIN_CONTROLLER_IP_ADDRESS"
port="{jboss.domain.master.port:9990}" security-realm="ManagementRealm"/>
</domain-controller>
```

For more information, see [Host Controller Configuration](#).

4. Configure authentication for each slave host.

Each slave server needs a username and password created in the domain controller's or standalone master's `ManagementRealm`. On the domain controller or standalone master, run the **`EAP_HOME/bin/add-user.sh`** command for each host. Add a management user for each host with the username that matches the host name of the slave.

Be sure to answer **yes** to the last question, that asks "Is this new user going to be used for one AS process to connect to another AS process?", so that you are provided with a secret value.

Example: `add-user.sh` Script Output (*trimmed*)

```
$ EAP_HOME/bin/add-user.sh

What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a): a

Username : slave1
Password : changeme
Re-enter Password : changeme
What groups do you want this user to belong to? (Please enter a comma separated list, or
leave blank for none)[ ]:
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for
server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities definition <secret
value="SECRET_VALUE" />
```

Copy the Base64-encoded secret value provided from this output, **`SECRET_VALUE`**, which may be used in the next step.

For more information, see the [Adding a User to the Master Domain Controller](#) section of the JBoss EAP *How To Configure Server Security* guide.

5. Modify the slave host's security realm to use the new authentication.

You can specify the password by setting the secret value in the server configuration, getting the password from a credential store or vault, or passing the password as a system property.

- Specify the Base64-encoded password value in the server configuration file using the Management CLI.

Use the following management CLI command to specify the secret value. Be sure to replace the **SECRET_VALUE** with the secret value returned from the add-user output from the previous step.

```
/host=SLAVE_HOST_NAME/core-service=management/security-  
realm=ManagementRealm/server-identity=secret:add(value="SECRET_VALUE")
```

You will need to reload the server. The **--host** argument is not applicable for a standalone server.

```
reload --host=HOST_NAME
```

For more information, see the [Configuring the Slave Controllers to Use the Credential](#) section of the JBoss EAP *How To Configure Server Security* guide.

- Configure the host to get the password from a credential store.
If you have stored the secret value in a credential store, you can use the following command to set the server secret to be a value from a credential store:

```
/host=SLAVE_HOST_NAME/core-service=management/security-  
realm=ManagementRealm/server-identity=secret:add(credential-reference=  
{store=STORE_NAME,alias=ALIAS})
```

You will need to reload the server. The **--host** argument is not applicable for a standalone server.

```
reload --host=HOST_NAME
```

For more information, see the [Credential Store](#) section of the JBoss EAP *How To Configure Server Security* guide.

- Configure the host to get the password from a vault.
 - a. Use the **EAP_HOME/bin/vault.sh** script to generate a masked password. It will generate a string in the format **VAULT::secret::password::VAULT_SECRET_VALUE**, for example:

```
VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNILWE4ODMtZjQ1NWNm  
NDU4ZDc1TEIORV9CUkVBS3ZhdWx0.
```



NOTE

When creating a password in the vault, it must be specified in plain text, not Base64-encoded.

- b. Use the following management CLI command to specify the secret value. Be sure to replace the **VAULT_SECRET_VALUE** with the masked password generated in the previous step.

```
/host=master/core-service=management/security-realm=ManagementRealm/server-  
identity=secret:add(value="${VAULT::secret::password::VAULT_SECRET_VALUE}")
```

You will need to reload the server. The **--host** argument is not applicable for a standalone server.

```
reload --host=HOST_NAME
```

For more information, see the [Password Vault](#) section of the JBoss EAP *How To Configure Server Security* guide.

- Specify the password as a system property.
The following examples use **server.identity.password** as the system property name for the password.
 - a. Specify the system property for the password in the server configuration file.
Use the following management CLI command to configure the secret identity to use the system property.

```
/host=SLAVE_HOST_NAME/core-service=management/security-  
realm=ManagementRealm/server-  
identity=secret:add(value="${server.identity.password}")
```

You will need to reload the server. The **--host** argument is not applicable for a standalone server.

```
reload --host=master
```

- b. Set the password for the system property when starting the server.
You can set the **server.identity.password** system property by passing it as a command line argument or in a properties file.
 - i. Pass in as a plain text command line argument.
Start the server and pass in the **server.identity.password** property.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -  
Dserver.identity.password=changeme
```



WARNING

The password must be entered in plain text and will be visible to anyone who issues a **ps -ef** command.

- ii. Set the property in a properties file.
Create a properties file and add the key/value pair to a properties file, for example:

```
server.identity.password=changeme
```

**WARNING**

The password is in plain text and will be visible to anyone who has access to this properties file.

Start the server with the command line arguments.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --
properties=PATH_TO_PROPERTIES_FILE
```

6. Restart the server.

The slave will now authenticate to the master using its host name as the username and the encrypted string as its password.

Your standalone server, or servers within a server group of a managed domain, are now configured as `mod_cluster` worker nodes. If you deploy a clustered application, its sessions are replicated to all cluster nodes for failover, and it can accept requests from an external web server or load balancer. Each node of the cluster discovers the other nodes using automatic discovery, by default.

24.6.4. Configure the `mod_cluster fail_on_status` Parameter

The **`fail_on_status`** parameter lists those HTTP status codes which, when returned by a worker node in a cluster, will mark that node as having failed. The load balancer will then send future requests to another worker node in the cluster. The failed worker node will remain in a **NOTOK** state until it sends the load balancer a **STATUS** message.

The **`fail_on_status`** parameter must be configured in the **`httpd`** configuration file of your load balancer. Multiple HTTP status codes for **`fail_on_status`** can be specified as a comma-separated list. The following example specifies the HTTP status codes **203** and **204** for **`fail_on_status`**.

Example: Configuring `fail_on_status`

```
ProxyPass / balancer://MyBalancer stickysession=JSESSIONID||jsessionid nofailover=on
failonstatus=203,204
ProxyPassReverse / balancer://MyBalancer
ProxyPreserveHost on
```

24.6.5. Migrate Traffic Between Clusters

After creating a new cluster using JBoss EAP, you can migrate traffic from the previous cluster to the new one as part of an upgrade process. In this task, you will see the strategy that can be used to migrate this traffic with minimal outage or downtime.

- A new cluster setup. We will call this cluster *ClusterNEW*.
- An old cluster setup that is being made redundant. We will call this cluster *ClusterOLD*.

Upgrade Process for Clusters - Load-Balancing Groups

1. Set up your new cluster using the steps described in the prerequisites.

2. In both *ClusterNEW* and *ClusterOLD*, ensure that the configuration option **sticky-session** is set to the default setting of **true**. Enabling this option means that all new requests made to a cluster node in any of the clusters will continue to go to the respective cluster node.

```
/profile=full-ha/subsystem=modcluster/proxy=default:write-attribute(name=sticky-session,value=true)
```

3. Set the **load-balancing-group** to **ClusterOLD**, assuming that all the cluster nodes in *ClusterOLD* are members of the *ClusterOLD* load-balancing group.

```
/profile=full-ha/subsystem=modcluster/proxy=default:write-attribute(name=load-balancing-group,value=ClusterOLD)
```

4. Add the nodes in *ClusterNEW* to the `mod_cluster` configuration individually using the process described in the [Configure a mod_cluster Worker Node](#) section. Additionally, use the aforementioned procedure and set their load-balancing group to *ClusterNEW*.
At this point, you can see an output similar to the undermentioned shortened example on the *mod_cluster-manager* console:

```
mod_cluster/<version>

LBGroup ClusterOLD: [Enable Nodes] [Disable Nodes] [Stop Nodes]
Node node-1-jvmroute (ajp://node1.oldcluster.example:8009):
  [Enable Contexts] [Disable Contexts] [Stop Contexts]
  Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
  Virtual Host 1:
    Contexts:
      /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

Node node-2-jvmroute (ajp://node2.oldcluster.example:8009):
  [Enable Contexts] [Disable Contexts] [Stop Contexts]
  Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
  Virtual Host 1:
    Contexts:
      /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

LBGroup ClusterNEW: [Enable Nodes] [Disable Nodes] [Stop Nodes]
Node node-3-jvmroute (ajp://node3.newcluster.example:8009):
  [Enable Contexts] [Disable Contexts] [Stop Contexts]
  Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
  Virtual Host 1:
    Contexts:
      /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

Node node-4-jvmroute (ajp://node4.newcluster.example:8009):
  [Enable Contexts] [Disable Contexts] [Stop Contexts]
  Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
  Virtual Host 1:
    Contexts:
      /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]
```

- Any old active sessions are within the *ClusterOLD* group, and any new sessions are created either within the *ClusterOLD* or *ClusterNEW* group. Next, we want to disable the whole *ClusterOLD* group, so that its cluster nodes may be removed without causing any error to currently active client's sessions.
Click on the **Disable Nodes** link for LBGroup *ClusterOLD* on the *mod_cluster-manager* web console.

From this point on, only requests belonging to already established sessions will be routed to members of the *ClusterOLD* load-balancing group. Any new client's sessions will be created in the *ClusterNEW* group only. As soon as there are no active sessions within *ClusterOLD* group, we can safely remove its members.



NOTE

Using **Stop Nodes** would command the load balancer to stop routing any requests to this domain immediately. This will force a failover to another load-balancing group which will cause session data loss to clients, provided there is no session replication between *ClusterNEW* and *ClusterOLD*.

Default Load-Balancing Group

In case the current *ClusterOLD* setup does not contain any load-balancing group settings, found from LBGroup: on the *mod_cluster-manager* console, one can still take advantage of disabling the *ClusterOLD* nodes. In this case, click on **Disable Contexts** for each of the *ClusterOLD* nodes. Contexts of these nodes will be disabled, and once there are no active sessions present they will be ready for removal. New clients' sessions will be created only on nodes with enabled contexts, presumably *ClusterNEW* members in this example.

Using the Management CLI

In addition to using the *mod_cluster-manager* web console, you can use the JBoss EAP management CLI to stop or disable a particular context.

Stop a Context

```
/host=master/server=server-one/subsystem=modcluster:stop-context(context=/my-deployed-application-context, virtualhost=default-host, waittime=0)
```

Stopping a context with **waittime** set to **0**, meaning no timeout, instructs the balancer to stop routing any request to it immediately, which forces failover to another available context.

If you set a timeout value using the **waittime** argument, no new sessions are created on this context, but existing sessions will continue to be directed to this node until they complete or the specified timeout has elapsed. The **waittime** argument defaults to **10** seconds.

Disable a Context

```
/host=master/server=server-one/subsystem=modcluster:disable-context(context=/my-deployed-application-context, virtualhost=default-host)
```

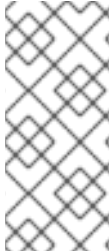
Disabling a context tells the balancer that no new sessions should be created on this context.

24.7. APACHE MOD_JK HTTP CONNECTOR

Apache mod_jk is an HTTP connector that is provided for customers who need it for compatibility purposes.

JBoss EAP can accept workloads from an Apache HTTP proxy server. The proxy server accepts client requests from the web front end, and passes the work to participating JBoss EAP servers. If sticky sessions are enabled, the same client request always goes to the same JBoss EAP server, unless the server is unavailable.

`mod_jk` communicates over the AJP 1.3 protocol. Other protocols can be used with `mod_cluster` or `mod_proxy`. See the [Overview of HTTP Connectors](#) for more information.



NOTE

`mod_cluster` is a more advanced load balancer than `mod_jk` and is the recommended HTTP connector. `mod_cluster` provides all of the functionality of `mod_jk`, plus additional features. Unlike the JBoss EAP `mod_cluster` HTTP connector, an Apache `mod_jk` HTTP connector does not know the status of deployments on servers or server groups, and cannot adapt where it sends its work accordingly.

See the [Apache `mod_jk` documentation](#) for more information.

24.7.1. Configure `mod_jk` in Apache HTTP Server

The `mod_jk` module, **`mod_jk.so`**, is already included when installing JBoss Core Services Apache HTTP Server or using JBoss Web Server; however, it is not loaded by default.



NOTE

Apache HTTP Server is no longer distributed with JBoss Web Server as of version 3.1.0.

Use the following steps to load and configure `mod_jk` in Apache HTTP Server. Note that these steps assume that you have already navigated to the **`httpd/`** directory for Apache HTTP Server, which will vary depending on your platform. For more information, see the installation instructions for your platform in the JBoss Core Services [Apache HTTP Server Installation Guide](#).



NOTE

Red Hat customers can also use the [Load Balancer Configuration Tool](#) on the Red Hat Customer Portal to quickly generate optimal configuration templates for `mod_jk` and other connectors. Note that you must be logged in to access this tool.

1. Configure the `mod_jk` module.



NOTE

A sample `mod_jk` configuration file is provided at **`conf.d/mod_jk.conf.sample`**. You can use this sample instead of creating your own file by removing the **`.sample`** extension and modifying its contents as needed.

Create a new file called **`conf.d/mod_jk.conf`**. Add the following configuration to the file, making sure to modify the contents to suite your needs.

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so
```

```

# Where to find workers.properties
JkWorkersFile conf.d/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Require ip 127.0.0.1
</Location>

```



NOTE

The JkMount directive specifies which URLs Apache HTTP Server must forward to the mod_jk module. Based on the directive's configuration, mod_jk sends the received URL to the correct workers. To serve static content directly and only use the load balancer for Java applications, the URL path must be **/application/***. To use mod_jk as a load balancer, use the value **/***, to forward all URLs to mod_jk.

Aside from general mod_jk configuration, this file specifies to load the **mod_jk.so** module, and defines where to find the **workers.properties** file.

2. Configure the mod_jk worker nodes.



NOTE

A sample workers configuration file is provided at **conf.d/workers.properties.sample**. You can use this sample instead of creating your own file by removing the **.sample** extension and modifying its contents as needed.

Create a new file called **conf.d/workers.properties**. Add the following configuration to the file, making sure to modify the contents to suite your needs.

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

For details on the syntax of the mod_jk **workers.properties** file and other advanced configuration options, see [mod_jk Worker Properties](#).

3. Optionally, specify a JKMountFile directive.
In addition to the JKMount directive in the **mod-jk.conf**, you can specify a file which contains multiple URL patterns to be forwarded to mod_jk.
 - a. Create a **uriworkermap.properties** file.



NOTE

A sample URI worker map configuration file is provided at **conf.d/uriworkermap.properties.sample**. You can use this sample instead of creating your own file by removing the **.sample** extension and modifying its contents as needed.

Create a new file called **conf.d/uriworkermap.properties**. Add a line for each URL pattern to be matched, for example:

```
# Simple worker configuration file
/*=loadbalancer
```

- b. Update the configuration to point to **uriworkermap.properties** file.

Append the following to **conf.d/mod_jk.conf**.

```
# Use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf.d/uriworkermap.properties
```

For more details on configuring mod_jk, see the [Configuring Apache HTTP Server to Load mod_jk](#) section of the JBoss Web Server *HTTP Connectors and Load Balancing Guide*.

24.7.2. Configure JBoss EAP to Communicate with mod_jk

The JBoss EAP **undertow** subsystem needs to specify a listener in order to accept requests from and send replies back to an external web server. Since mod_jk uses the AJP protocol, an AJP listener must be configured.

If you are using one of the default high availability configurations, *ha* or *full-ha*, an AJP listener is already configured.

For instructions, see [Accepting Requests From External Web Servers](#).

24.8. APACHE MOD_PROXY HTTP CONNECTOR

Apache mod_proxy is an HTTP connector that supports connections over AJP, HTTP, and HTTPS protocols. mod_proxy can be configured in load-balanced or non-load-balanced configurations, and it supports the notion of sticky sessions.

The mod_proxy module requires JBoss EAP to have the HTTP, HTTPS or AJP listener configured in the **undertow** subsystem, depending on which protocol you plan to use.



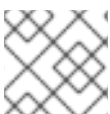
NOTE

mod_cluster is a more advanced load balancer than mod_proxy and is the recommended HTTP connector. mod_cluster provides all of the functionality of mod_proxy, plus additional features. Unlike the JBoss EAP mod_cluster HTTP connector, an Apache mod_proxy HTTP connector does not know the status of deployments on servers or server groups, and cannot adapt where it sends its work accordingly.

See the [Apache mod_proxy documentation](#) for more information.

24.8.1. Configure mod_proxy in Apache HTTP Server

The mod_proxy modules are already included when installing JBoss Core Services Apache HTTP Server or using JBoss Web Server and are loaded by default.



NOTE

Apache HTTP Server is no longer distributed with JBoss Web Server as of version 3.1.0.

See the appropriate section below to configure a basic [load-balancing](#) or [non-load-balancing](#) proxy. These steps assume that you have already navigated to the **httpd/** directory for Apache HTTP Server, which will vary depending on your platform. For more information, see the installation instructions for

your platform in the JBoss Core Services [Apache HTTP Server Installation Guide](#). These steps also assume that the necessary HTTP listener has already been configured in the JBoss EAP **undertow** subsystem.



NOTE

Red Hat customers can also use the [Load Balancer Configuration Tool](#) on the Red Hat Customer Portal to quickly generate optimal configuration templates for `mod_proxy` and other connectors. Note that you must be logged in to access this tool.

Add a Non-load-balancing Proxy

Add the following configuration to your **conf/httpd.conf** file, directly beneath any other **<VirtualHost>** directives you may have. Replace the values with ones appropriate to your setup.

```
<VirtualHost *:80>
# Your domain name
ServerName YOUR_DOMAIN_NAME

ProxyPreserveHost On

# The IP and port of JBoss
# These represent the default values, if your httpd is on the same host
# as your JBoss managed domain or server

ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>
</VirtualHost>
```

Add a Load-balancing Proxy



NOTE

The default Apache HTTP Server configuration has the **mod_proxy_balancer.so** module disabled, as it is incompatible with `mod_cluster`. In order to complete this task, you will need to load this module and disable the `mod_cluster` modules.

To use `mod_proxy` as a load balancer, and send work to multiple JBoss EAP instances, add the following configuration to your **conf/httpd.conf** file. The example IP addresses are fictional. Replace them with the appropriate values for your environment.

```
<Proxy balancer://mycluster>

Order deny,allow
Allow from all

# Add each JBoss Enterprise Application Server by IP address and port.
```

```
# If the route values are unique like this, one node will not fail over to the other.
BalancerMember http://192.168.1.1:8080 route=node1
BalancerMember http://192.168.1.2:8180 route=node2
</Proxy>

<VirtualHost *:80>
# Your domain name
ServerName YOUR_DOMAIN_NAME

ProxyPreserveHost On
ProxyPass / balancer://mycluster/

# The location of the HTML files, and access control information DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>

</VirtualHost>
```

The examples above all communicate using the HTTP protocol. You can use AJP or HTTPS protocols instead, if you load the appropriate `mod_proxy` modules. See the [Apache mod_proxy documentation](#) for more details.

Enable Sticky Sessions

A *sticky session* means that if a client request originally goes to a specific JBoss EAP worker, all future requests will be sent to the same worker, unless it becomes unavailable. This is almost always the recommended behavior.

To enable sticky sessions for `mod_proxy`, add the **stickysession** parameter to the **ProxyPass** statement.

```
ProxyPass / balancer://mycluster stickysession=JSESSIONID
```

You can specify additional parameters to the **ProxyPass** statement, such as **lbmethod** and **nofailover**. See the [Apache mod_proxy documentation](#) for more information on the available parameters.

24.8.2. Configure JBoss EAP to Communicate with mod_proxy

The JBoss EAP **undertow** subsystem needs to specify a listener in order to accept requests from and send replies back to an external web server. Depending on the protocol that you will be using, you may need to configure a listener.

An HTTP listener is configured in the JBoss EAP default configuration. If you are using one of the default high availability configurations, *ha* or *full-ha*, an AJP listener is also preconfigured.

For instructions, see [Accepting Requests From External Web Servers](#).

24.9. MICROSOFT ISAPI CONNECTOR

The Internet Server API (ISAPI) is a set of APIs used to write OLE Server extensions and filters for web servers such as Microsoft's Internet Information Services (IIS). **isapi_redirect.dll** is an extension of *mod_jk* adjusted to IIS. **isapi_redirect.dll** enables you to configure JBoss EAP instances as worker nodes with IIS as a load balancer.

**NOTE**

See [JBoss EAP supported configurations](#) for information on the supported configurations of Windows Server and IIS.

24.9.1. Configure Microsoft IIS to Use the ISAPI Connector

Download the ISAPI connector from the Red Hat Customer Portal:

1. Open a browser and log in to the Red Hat Customer Portal [JBoss Software Downloads page](#).
2. Select **Web Connectors** in the **Product** drop-down menu.
3. Select the latest JBoss Core Services version from the **Version** drop-down menu.
4. Find the **Red Hat JBoss Core Services ISAPI Connector** in the list, and click the **Download** link.
5. Extract the archive and copy the contents of the **sbin** directory to a location on your server. The instructions below assume that the contents were copied to **C:\connectors**.

To configure the IIS Redirector Using the IIS Manager (IIS 7):

1. Open the IIS manager by clicking **Start → Run**, and typing **inetmgr**.
2. In the tree view pane at the left, expand **IIS 7**.
3. Double-click **ISAPI and CGI Registrations** to open it in a new window.
4. In the **Actions** pane, click **Add**. The **Add ISAPI or CGI Restriction** window opens.
5. Specify the following values:
 - **ISAPI or CGI Path** **C:\connectors\isapi_redirect.dll**
 - **Description**: **jboss**
 - **Allow extension path to execute** select the check box.
6. Click **OK** to close the **Add ISAPI or CGI Restriction** window.
7. Define a JBoss Native virtual directory
 - Right-click *Default Web Site*, and click *Add Virtual Directory*. The *Add Virtual Directory* window opens.
 - Specify the following values to add a virtual directory:
 - **Alias**: **jboss**
 - **Physical Path** **C:\connectors**
 - Click **OK** to save the values and close the **Add Virtual Directory** window.
8. Define a JBoss Native ISAPI Redirect Filter
 - In the tree view pane, expand **Sites → Default Web Site**.
 - Double-click **ISAPI Filters**. The **ISAPI Filters Features** view appears.

- In the **Actions** pane, click **Add**. The **Add ISAPI Filter** window appears.
 - Specify the following values in the **Add ISAPI Filter** window:
 - **Filter name:** **jboss**
 - **Executable:** **C:\connectors\isapi_redirect.dll**
 - Click **OK** to save the values and close the **Add ISAPI Filters** window.
9. Enable the ISAPI-dll handler
- Double-click the **IIS 7** item in the tree view pane. The **IIS 7 Home Features View** opens.
 - Double-click **Handler Mappings**. The **Handler Mappings Features View** appears.
 - In the **Group by** combo box, select **State**. The **Handler Mappings** are displayed in **Enabled and Disabled Groups**.
 - Find **ISAPI-dll**. If it is in the **Disabled** group, right-click it and select **Edit Feature Permissions**.
 - Enable the following permissions:
 - **Read**
 - **Script**
 - **Execute**
 - Click **OK** to save the values, and close the **Edit Feature Permissions** window.

Microsoft IIS is now configured to use the ISAPI connector.

24.9.2. Configure the ISAPI Connector to Send Client Requests to JBoss EAP

This task configures a group of JBoss EAP servers to accept requests from the ISAPI connector. It does not include configuration for load-balancing or high-availability failover.

This configuration is done on the IIS server, and assumes that you already have JBoss EAP configured to [accept requests from an external web server](#) . You also need full administrator access to the IIS server and to have [configured IIS to use the ISAPI connector](#) .

Create Property Files and Set Up Redirection

1. Create a directory to store logs, property files, and lock files.
The rest of this procedure assumes that you are using the directory **C:\connectors** for this purpose. If you use a different directory, modify the instructions accordingly.
2. Create the **isapi_redirect.properties** file.
Create a new file called **C:\connectors\isapi_redirect.properties**. Copy the following contents into the file.

```
# Configuration file for the ISAPI Connector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll
```

```
# Full path to the log file for the ISAPI Connector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
log_level=info

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermmap.properties file
worker_mount_file=c:\connectors\uriworkermmap.properties

#Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

If you do not want to use a **rewrite.properties** file, comment out the last line by placing a **#** character at the beginning of the line.

3. Create the **uriworkermmap.properties** file

The **uriworkermmap.properties** file contains mappings between deployed application URLs and which worker handles requests to them. The following example file shows the syntax of the file. Place your **uriworkermmap.properties** file into **C:\connectors**.

```
# images and css files for path /status are provided by worker01
/status=worker01
/images/*=worker01
/css/*=worker01

# Path /web-console is provided by worker02
# IIS (customized) error page is used for http errors with number greater or equal to 400
# css files are provided by worker01
/web-console/*=worker02;use_server_errors=400
/web-console/css/*=worker01

# Example of exclusion from mapping, logo.gif won't be displayed
# !/web-console/images/logo.gif=*

# Requests to /app-01 or /app-01/something will be routed to worker01
/app-01/*=worker01

# Requests to /app-02 or /app-02/something will be routed to worker02
/app-02/*=worker02
```

4. Create the **workers.properties** file.

The **workers.properties** file contains mapping definitions between worker labels and server instances. This file follows the syntax of the same file used for [Apache mod_jk worker properties](#) configuration.

The following is an example of a **workers.properties** file. The worker names, **worker01** and **worker02**, must match the **instance-id** [configured in the JBoss EAP undertow subsystem](#).

Place this file into the **C:\connectors** directory.

```
# An entry that lists all the workers defined
worker.list=worker01, worker02
```

```
# Entries that define the host and port associated with these workers

# First JBoss EAP server definition, port 8009 is standard port for AJP in EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second JBoss EAP server definition
worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. Create the **rewrite.properties** file.

The **rewrite.properties** file contains simple URL rewriting rules for specific applications. The rewritten path is specified using name-value pairs, as shown in the example below. Place this file into the **C:\connectors** directory.

```
#Simple example
# Images are accessible under abc path
/app-01/abc=/app-01/images/
```

6. Restart your IIS server by using the **net stop** and **net start** commands.

```
C:\> net stop was /Y
C:\> net start w3svc
```

The IIS server is configured to send client requests to the specific JBoss EAP servers you have configured, on an application-specific basis.

24.9.3. Configure the ISAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers

This configuration balances client requests across the JBoss EAP servers you specify. This configuration is done on the IIS server, and assumes that you already have JBoss EAP configured to [accept requests from an external web server](#). You also need full administrator access to the IIS server and to have [configured IIS to use the ISAPI connector](#).

Balance Client Requests Across Multiple Servers

1. Create a directory to store logs, property files, and lock files.
The rest of this procedure assumes that you are using the directory **C:\connectors** for this purpose. If you use a different directory, modify the instructions accordingly.
2. Create the **isapi_redirect.properties** file.
Create a new file called **C:\connectors\isapi_redirect.properties**. Copy the following contents into the file.

```
# Configuration file for the ISAPI Connector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Connector
log_file=c:\connectors\isapi_redirect.log
```

```
# Log level (debug, info, warn, error or trace)
log_level=info

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermmap.properties file
worker_mount_file=c:\connectors\uriworkermmap.properties

#OPTIONAL: Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

If you do not want to use a **rewrite.properties** file, comment out the last line by placing a **#** character at the beginning of the line.

3. Create the **uriworkermmap.properties** file.

The **uriworkermmap.properties** file contains mappings between deployed application URLs and which worker handles requests to them. The following example file shows the syntax of the file, with a load-balanced configuration. The wildcard (*) character sends all requests for various URL sub-directories to the load balancer called router. The configuration of the load balancer is covered in the next step.

Place your **uriworkermmap.properties** file into **C:\connectors**.

```
# images, css files, path /status and /web-console will be
# provided by nodes defined in the load-balancer called "router"
/css/*=router
/images/*=router
/status=router
/web-console/*=router

# Example of exclusion from mapping, logo.gif won't be displayed
# !/web-console/images/logo.gif=*

# Requests to /app-01 and /app-02 will be routed to nodes defined
# in the load-balancer called "router"
/app-01/*=router
/app-02/*=router

# mapping for management console, nodes in cluster can be enabled or disabled here
/jkmanager/*=status
```

4. Create the **workers.properties** file.

The **workers.properties** file contains mapping definitions between worker labels and server instances. This file follows the syntax of the same file used for [Apache mod_jk worker properties](#) configuration.

The following is an example of a **workers.properties** file. The load balancer is configured near the end of the file, to comprise workers **worker01** and **worker02**. These worker names must match the **instance-id** configured in the JBoss EAP [undertow](#) subsystem.

Place this file into the **C:\connectors** directory.

```
# The advanced router LB worker
```

```

worker.list=router,status

# First EAP server definition, port 8009 is standard port for AJP in EAP
#
# lbfactor defines how much the worker will be used.
# The higher the number, the more requests are served
# lbfactor is useful when one machine is more powerful
# ping_mode=A – all possible probes will be used to determine that
# connections are still working

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker for jkmanager
worker.status.type=status

```

5. Create the **rewrite.properties** file.

The **rewrite.properties** file contains simple URL rewriting rules for specific applications. The rewritten path is specified using name-value pairs, as shown in the example below. Place this file into the **C:\connectors** directory.

```

#Simple example
# Images are accessible under abc path
/app-01/abc/=/app-01/images/
Restart the IIS server.

Restart your IIS server by using the net stop and net start commands.
C:\> net stop was /Y
C:\> net start w3svc

```

The IIS server is configured to send client requests to the JBoss EAP servers referenced in the **workers.properties** file, spreading the load across the servers in a **1:3** ratio. This ratio is derived from the load-balancing factor, **lbfactor**, assigned to each server.

24.10. ORACLE NSAPI CONNECTOR

The Netscape Server API (NSAPI) is an API provided by Oracle iPlanet Web Server, formerly Netscape

Web Server, for implementing extensions to the server. These extensions are known as server plugins. The NSAPI connector is used in **nsapi_redirector.so**, which is an extension of mod_jk adjusted to Oracle iPlanet Web Server. The NSAPI connector enables you to configure JBoss EAP instances as worker nodes with Oracle iPlanet Web Server as a load balancer.



NOTE

See the [JBoss EAP supported configurations](#) for information on the supported configurations of Solaris and Oracle iPlanet Web Server.

24.10.1. Configure Oracle iPlanet Web Server to use the NSAPI Connector

Prerequisites

- JBoss EAP is installed and configured on each server that will serve as a worker.

Download the NSAPI connector from the Red Hat Customer Portal:

1. Open a browser and log in to the Red Hat Customer Portal [JBoss Software Downloads page](#).
2. Select **Web Connectors** in the **Product** drop-down menu.
3. Select the latest JBoss Core Services version from the **Version** drop-down menu.
4. Find the **Red Hat JBoss Core Services NSAPI Connector** in the list, ensuring that you select the correct platform and architecture for your system, and click the **Download** link.
5. Extract the **nsapi_redirector.so** file, which is located in either the **lib/** or the **lib64/** directory, into either the **IPLANET_CONFIG/lib/** or the **IPLANET_CONFIG/lib64/** directory.

Set up the NSAPI Connector:



NOTE

In these instructions, **IPLANET_CONFIG** refers to the Oracle iPlanet configuration directory, which is usually **/opt/oracle/webserver7/config/**. If your Oracle iPlanet configuration directory is different, modify the instructions accordingly.

1. Disable servlet mappings.
Open the **IPLANET_CONFIG/default.web.xml** file and locate the section with the heading *Built In Server Mappings*. Disable the mappings to the following three servlets, by wrapping them in XML comment characters (**<!--** and **-->**).
- default
 - invoker
 - jsp
- The following example configuration shows the disabled mappings.

```
<!-- ===== Built In Servlet Mappings ===== -->
<!-- The servlet mappings for the built in servlets defined above. -->
<!-- The mapping for the default servlet -->
<!--servlet-mapping>
```

```

<servlet-name>default</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping-->
<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
<servlet-name>invoker</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->
<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
<servlet-name>jsp</servlet-name>
<url-pattern>*.jsp</url-pattern>
</servlet-mapping-->

```

Save and exit the file.

2. Configure the iPlanet Web Server to load the NSAPI connector module.

Add the following lines to the end of the ***IPLANET_CONFIG/magnus.conf*** file, modifying file paths to suit your configuration. These lines define the location of the ***nsapi_redirector.so*** module, as well as the ***workers.properties*** file, which lists the workers and their properties.

```

Init fn="load-modules" funcs="jk_init,jk_service" shlib="/lib/nsapi_redirector.so" shlib_flags="
(global|now)"

Init fn="jk_init" worker_file="IPLANET_CONFIG/connectors/workers.properties"
log_level="info" log_file="IPLANET_CONFIG/connectors/nsapi.log"
shm_file="IPLANET_CONFIG/connectors/tmp/jk_shm"

```

The configuration above is for a 32-bit architecture. If you use 64-bit Solaris, change the string ***lib/nsapi_redirector.so*** to ***lib64/nsapi_redirector.so***.

Save and exit the file.

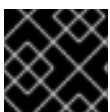
3. Configure the NSAPI connector.

You can configure the NSAPI connector for a basic configuration, with no load balancing, or a load-balancing configuration. Choose one of the following options, after which your configuration will be complete.

- [Configure the NSAPI Connector to Send Client Requests to JBoss EAP](#)
- [Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers](#)

24.10.2. Configure the NSAPI Connector to Send Client Requests to JBoss EAP

This task configures the NSAPI connector to redirect client requests to JBoss EAP servers with no load balancing or failover. The redirection is done on a per-deployment, and therefore per-URL, basis.



IMPORTANT

You must have already [configured the NSAPI connector](#) before continuing with this task.

Set Up the Basic HTTP Connector

1. Define the URL paths to redirect to the JBoss EAP servers.



NOTE

In ***IPLANET_CONFIG/obj.conf***, spaces are not allowed at the beginning of a line, except when the line is a continuation of the previous line.

Edit the ***IPLANET_CONFIG/obj.conf*** file. Locate the section which starts with `<Object name="default">`, and add each URL pattern to match, in the format shown by the example file below. The string *jknsapi* refers to the HTTP connector which will be defined in the next step. The example shows the use of wildcards for pattern matching.

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(/*)" name="jknsapi"
</Object>
```

2. Define the worker which serves each path.

Continue editing the ***IPLANET_CONFIG/obj.conf*** file. Add the following directly after the closing tag of the section you have just finished editing: `</Object>`.

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="worker01" path="/status"
Service fn="jk_service" worker="worker02" path="/nc(/*)"
Service fn="jk_service" worker="worker01"
</Object>
```

The example above redirects requests to the URL path */status* to the worker called *worker01*, and all URL paths beneath ***/nc/*** to the worker called *worker02*. The third line indicates that all URLs assigned to the *jknsapi* object which are not matched by the previous lines are served to *worker01*.

Save and exit the file.

3. Define the workers and their attributes.

Create a file called ***workers.properties*** in the ***IPLANET_CONFIG/connectors/*** directory. Paste the following contents into the file, and modify them to suit your environment.

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

The ***workers.properties*** file uses the same syntax as Apache `mod_jk`.

Save and exit the file.

- Restart the iPlanet Web Server

Issue the following command to restart the iPlanet Web Server.

```
IPLANET_CONFIG/./bin/stopserv
IPLANET_CONFIG/./bin/startserv
```

iPlanet Web Server now sends client requests to the URLs you have configured to deployments on JBoss EAP.

24.10.3. Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers

This task configures the NSAPI connector to send client requests to JBoss EAP servers in a load-balancing configuration.



IMPORTANT

You must have already [configured the NSAPI connector](#) before continuing with this task.

Configure the Connector for Load Balancing

- Define the URL paths to redirect to the JBoss EAP servers.



NOTE

In ***IPLANET_CONFIG/obj.conf***, spaces are not allowed at the beginning of a line, except when the line is a continuation of the previous line.

Edit the ***IPLANET_CONFIG/obj.conf*** file. Locate the section which starts with **<Object name="default">**, and add each URL pattern to match, in the format shown by the example file below. The string **jknsapi** refers to the HTTP connector that will be defined in the next step. The example shows the use of wildcards for pattern matching.

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jkmanager/*" name="jknsapi"
</Object>
```

- Define the worker that serves each path.

Continue editing the ***IPLANET_CONFIG/obj.conf*** file. Directly after the closing tag for the section you modified in the previous step, **</Object>**, add the following new section and modify it to your needs:

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="status" path="/jkmanager(/*)"
```

```
Service fn="jk_service" worker="router"
</Object>
```

This **jksnapi** object defines the worker nodes used to serve each path that was mapped to the **name="jksnapi"** mapping in the **default** object. Everything except for URLs matching **/jkmanager/*** is redirected to the worker called **router**.

3. Define the workers and their attributes.

Create a file called **workers.properties** in **IPLANET_CONFIG/connector/**. Paste the following contents into the file, and modify them to suit your environment.

```
# The advanced router LB worker
# A list of each worker
worker.list=router,status

# First JBoss EAP server
# (worker node) definition.
# Port 8009 is the standard port for AJP
#

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second JBoss EAP server
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the load-balancer called "router"
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker
worker.status.type=status
```

The **workers.properties** file uses the same syntax as Apache `mod_jk`.

Save and exit the file.

4. Restart the iPlanet Web Server 7.0.

```
IPLANET_CONFIG/..bin/stopserv
IPLANET_CONFIG/..bin/startserv
```

The iPlanet Web Server redirects the URL patterns you have configured to your JBoss EAP servers in a load-balancing configuration.

CHAPTER 25. ECLIPSE MICROPROFILE

25.1. USING ECLIPSE MICROPROFILE CONFIG TO MANAGE CONFIGURATION



IMPORTANT

Eclipse Microprofile Config is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

25.1.1. About the MicroProfile Config SmallRye Subsystem

Configuration data can change dynamically and applications need to be able to access the latest configuration information without restarting the server. [Eclipse MicroProfile Config](#) provides portable externalization of configuration data. This allows applications and microservices to be configured to run in multiple environments without a need for modification or repackaging. Eclipse MicroProfile Config functionality is implemented in JBoss EAP using the [SmallRye Config](#) component and is provided by the **microprofile-config-smallrye** subsystem. This subsystem is included in the default JBoss EAP 7.3 configuration.

25.1.2. ConfigSources Supported by the MicroProfile Config SmallRye Subsystem

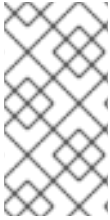
MicroProfile Config configuration properties, which can be in different formats and can come from different locations, are provided by ConfigSources, which are implementations of the [org.eclipse.microprofile.config.spi.ConfigSource](#) interface.

The MicroProfile Config specification provides the following default **ConfigSource** implementations for retrieving configuration values.

- Using **System.getProperties()**.
- Using **System.getenv()**.
- From all **META-INF/microprofile-config.properties** files on the class path.

The **microprofile-config-smallrye** subsystem supports the following additional types of **ConfigSource** resources for retrieving configuration values.

- From [properties](#).
- From a [files in a directory](#).
- From a [ConfigSource](#) class.
- From a [ConfigSourceProvider](#) class.



NOTE

Although the sections below use the management CLI to configure the **microprofile-config-smallrye** subsystem, you can also use the management console to accomplish these tasks by navigating to **Configuration → Subsystems → MicroProfile Config →** and clicking **View**.

See [MicroProfile Config SmallRye Subsystem Attributes](#) for the list of attributes available for configuring the MicroProfile Config SmallRye subsystem.

Obtaining ConfigSource Configuraton from Properties

You can store properties directly for access by a **ConfigSource** by adding a **config-source** attribute and specifying the **properties** as a list.

The following management CLI command creates a **ConfigSource** containing configuration data for the **property1** and **property2** properties.

```
/subsystem=microprofile-config-smallrye/config-source=props:add(properties={property1=value1,property2=value2})
```

This command results in the following XML configuration for the **microprofile-config-smallrye** subsystem.

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
  <config-source name="props">
    <property name="property1" value="value1"/>
    <property name="property2" value="value2"/>
  </config-source>
</subsystem>
```

Obtaining ConfigSource Configuraton from a Directory

You can create a **ConfigSource** to read properties from files in a directory by adding the **config-source** attribute and specifying the directory that contains the files. The name of each file in the **dir** directory is the name of a property and the file content is the value of the property,

The following management CLI command creates a **ConfigSource** that reads configuration properties from files in the **/etc/config/numbers-app/** directory.

```
/subsystem=microprofile-config-smallrye/config-source=file-props:add(dir={path=/etc/config/numbers-app})
```

This command results in the following XML configuration for the **microprofile-config-smallrye** subsystem.

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
  <config-source name="file-props">
    <dir path="/etc/config/numbers-app"/>
  </config-source>
</subsystem>
```

This structure corresponds to the structure used by OpenShift **ConfigMaps**. The **dir** value corresponds to the **mountPath** in the **ConfigMap** definition in OpenShift or Kubernetes.

Any application deployed in JBoss EAP can programmatically access the properties that are stored in the directory.

Assume that the directory `/etc/config/numbers-app/` contains the following two files:

- **num.size**: This file contains the value **5**.
- **num.max**: This file contains the value **100**.

In the code example below, **num.size** returns **5** and **num.max** returns **100**.

```
@Inject
@ConfigProperty(name = "num.size")
int numSize;

@Inject
@ConfigProperty(name = "num.max")
int numMax;
```

Obtaining ConfigSource Configuraton from a ConfigSource Class

You can create and configure a custom [org.eclipse.microprofile.config.spi.ConfigSource](#) implementation class to provide a source for the configuration values.

The following management CLI command creates a **ConfigSource** for the implementation class named **org.example.MyConfigSource** that is provided by a JBoss Module named **org.example**.

```
/subsystem=microprofile-config-smallrye/config-source=my-config-source:add(class=
{name=org.example.MyConfigSource, module=org.example})
```

This command results in the following XML configuration for the **microprofile-config-smallrye** subsystem.

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
  <config-source name="my-config-source">
    <class name="org.example.MyConfigSource" module="org.example"/>
  </config-source>
</subsystem>
```

Properties provided by this **ConfigSource** class are available to any JBoss EAP deployment.

For information about how to add a global module to the JBoss EAP server, see [Define Global Modules](#).

Obtaining ConfigSource Configuraton from a ConfigSourceProvider Class

You can create and configure a custom [org.eclipse.microprofile.config.spi.ConfigSourceProvider](#) implementation class that registers implementations for multiple **ConfigSource** instances.

The following management CLI command creates a **config-source-provider** for the implementation class named **org.example.MyConfigSourceProvider** that is provided by a JBoss Module named **org.example**.

```
/subsystem=microprofile-config-smallrye/config-source-provider=my-config-source-
provider:add(class={name=org.example.MyConfigSourceProvider, module=org.example})
```

This command results in the following XML configuration for the **microprofile-config-smallrye** subsystem.

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
  <config-source-provider name="my-config-source-provider">
    <class name="org.example.MyConfigSourceProvider" module="org.example"/>
  </config-source-provider>
</subsystem>
```

Properties provided by the **ConfigSourceProvider** implementation are available to any JBoss EAP deployment.

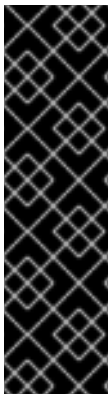
For information about how to add a global module to the JBoss EAP server, see [Define Global Modules](#).

25.1.3. Deploying Applications that Access ConfigSources

Applications that access MicroProfile Config in their Java code must enable CDI, either by including a **META-INF/beans.xml** or **WEB-INF/beans.xml** file in the deployment archive, or by including a CDI bean annotation.

For more information about CDI, see [Contexts and Dependency Injection \(CDI\)](#) in the JBoss EAP *Development Guide*.

25.2. TRACING REQUESTS WITH THE MICROPROFILE OPENTRACING SMALLRYE SUBSYSTEM



IMPORTANT

Eclipse Microprofile OpenTracing is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

25.2.1. About Eclipse MicroProfile OpenTracing

The ability to trace requests across service boundaries is important, especially in a microservices environment where a request can flow through multiple services during its life cycle. The [Eclipse Microprofile OpenTracing](#) specification defines behaviors and an API for accessing an OpenTracing compliant **Tracer** object within a Jakarta RESTful Web Services application. The behaviors specify how OpenTracing Spans are to be created automatically for incoming and outgoing requests. The API defines how to explicitly disable or enable tracing for given endpoints.

25.2.2. About the MicroProfile OpenTracing SmallRye Subsystem

Eclipse Microprofile OpenTracing functionality is implemented in JBoss EAP using the [SmallRye OpenTracing](#) component and is provided by the **microprofile-opentracing-smallrye** subsystem. This subsystem implements Microprofile 1.3.0, which includes support for tracing requests to JAX-RS endpoints and CDI beans and is included in the default JBoss EAP 7.3 configuration.

The **microprofile-opentracing-smallrye** subsystem ships with the Jaeger Client as the default tracer, plus a set of instrumentation libraries for components commonly used in Jakarta EE applications, such

as Jakarta RESTful Web Services and Contexts and Dependency Injection. Each individual WAR deployed to the JBoss EAP server automatically has its own **Tracer** instance. Each WAR within an EAR is treated as an individual WAR, and each has its own **Tracer** instance. By default, the service name used with the Jaeger Client is derived from the deployment's name, which is usually the WAR file name.

25.2.3. Configuring the MicroProfile OpenTracing SmallRye Subsystem

The **microprofile-opentracing-smallrye** subsystem is included in the default JBoss EAP 7.3 configuration. Because there can be a memory or performance costs associated with OpenTracing enabled, you might want to disable this subsystem.

Use the following management CLI commands to disable the MicroProfile OpenTracing feature globally for the server instance by removing the subsystem from the server configuration.

1. Remove the subsystem.

```
/subsystem=microprofile-opentracing-smallrye:remove()
```

2. Reload the server for the changes to take effect.

```
reload
```

Use the following management CLI commands to enable the MicroProfile OpenTracing feature globally for the server instance by adding the subsystem to the server configuration.

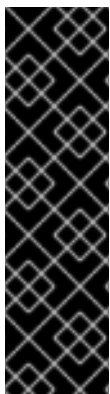
1. Add the subsystem.

```
/subsystem=microprofile-opentracing-smallrye:add()
```

2. Reload the server for the changes to take effect.

```
reload
```

There are no other configuration options available for the **microprofile-opentracing-smallrye** subsystem. Instead, you configure the Jaeger Client by setting system properties or environment variables. See the [Jaeger documentation](#) for information about how to configure the Jaeger Client. See [Configuration via Environment](#) in the Jaeger documentation for the list of valid system properties.



IMPORTANT

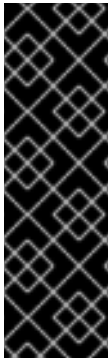
Because this feature is provided as Technology Preview, the current configuration options, particularly those related to configuring the Jaeger Client tracer using system properties and environment variables, might change in incompatible ways in future releases.

Also be aware that, by default, the Jaeger Client for Jakarta has a probabilistic sampling strategy that is set to **0.001**, meaning that only approximately one in one thousand traces will be sampled. To sample every request, set the system properties

JAEGER_SAMPLER_TYPE to **const** and **JAEGER_SAMPLER_PARAM** to **1**.

For information about how to override the default tracer and how to trace CDI beans, see [Using Eclipse MicroProfile OpenTracing to Trace Requests](#) in the *Development Guide*.

25.3. MONITOR SERVER HEALTH USING ECLIPSE MICROPROFILE HEALTH



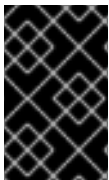
IMPORTANT

Eclipse MicroProfile Health is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

25.3.1. MicroProfile Health SmallRye Subsystem

JBoss EAP includes the [SmallRye Health](#) component, which provides [Eclipse MicroProfile Health](#) functionality using the **microprofile-health-smallrye** subsystem. This subsystem is used to determine whether the JBoss EAP instance is responding as expected, and is enabled by default.



IMPORTANT

By default, the MicroProfile Health SmallRye subsystem only evaluates whether the server is running. To include custom health checks, see the [Implement a Custom Health Check](#) section in the *Development Guide*.

Microprofile Health is only available when running JBoss EAP as a standalone server.

25.3.2. Health Check Using the Management CLI Examples

Health checks can be examined using the management CLI.

- The **:check** attribute demonstrates obtaining the current health of the server (**liveness** and **readiness** probes).

```
/subsystem=microprofile-health-smallrye:check
{
  "outcome" => "success",
  "result" => {
    "outcome" => "UP",
    "checks" => []
  }
}
```

- The **:check-live** attribute obtains the health data for **liveness** probes only.

```
/subsystem=microprofile-health-smallrye:check-live
{
  "outcome" => "success",
  "result" => {
    "outcome" => "UP",
  }
}
```

```

    "checks" => []
  }
}

```

- The **:check-ready** attribute obtains the health data for **readiness** probes only.

```

/subsystem=microprofile-health-smallrye:check-ready
{
  "outcome" => "success",
  "result" => {
    "outcome" => "UP",
    "checks" => []
  }
}

```

25.3.3. Examining the Health Check Using the Management Console

The management console includes a check runtime operation that shows the health checks and the global outcome as boolean value.

To obtain the current health status of the server from the management console:

1. Navigate to the **Runtime** tab and select the server.
2. In the **Monitor** column, click **MicroProfile Health** → **View**.

25.3.4. Examine the Health Check Using the HTTP Endpoint

Health check is automatically deployed to the **health** context on JBoss EAP.

That means that in addition to being able to examine it using the management CLI, you can also obtain the current health using the HTTP endpoint. The default address for the `/health` endpoint, accessible from the management interfaces, is `http://127.0.0.1:9990/health`.

1. To obtain the current health of the server using the HTTP endpoint, the following URL would be used.

```
http://HOST:9990/health
```

Accessing this context displays the health check in JSON format, indicating whether the server is healthy. This endpoint checks the health of both **liveness** and **readiness** probes.

1. The **/health/live** endpoint checks the current health of **liveness** probes.

```
http://HOST:9990/health/live
```

1. The **/health/ready** endpoint checks the current health of **readiness** probes.

```
http://HOST:9990/health/ready
```

25.3.5. Global Status When Probes are not Defined

The **:empty-readiness-checks-status** and **:empty-liveness-checks-status** are management

attributes that specify the global status when no **readiness** or **liveness** probes are defined. These attributes allow applications to report 'DOWN' until their probes verify that the application is ready/live. By default, these attributes will report 'UP'.

1. The **:empty-readiness-checks-status** attribute specifies the global status for **readiness** probes if no **readiness** probes have been defined.

```
/subsystem=microprofile-health-smallrye:read-attribute(name=empty-readiness-checks-status)
{
  "outcome" => "success",
  "result" => expression "${env.MP_HEALTH_EMPTY_READINESS_CHECKS_STATUS:UP}"
}
```

1. The **:empty-liveness-checks-status** attribute specifies the global status for **liveness** probes if no **liveness** probes have been defined.

```
/subsystem=microprofile-health-smallrye:read-attribute(name=empty-liveness-checks-status)
{
  "outcome" => "success",
  "result" => expression "${env.MP_HEALTH_EMPTY_LIVENESS_CHECKS_STATUS:UP}"
}
```

The **/health** HTTP endpoint and **:check** operation that check both **readiness** and **liveness** probes also take into account these attributes.

25.3.6. Enable Authentication for the HTTP Endpoint

The **health** context can be configured to require that users be authorized to access the context.

To enable authentication on this endpoint:

1. Set the **security-enabled** attribute to **true** on the **microprofile-health-smallrye** subsystem.

```
/subsystem=microprofile-health-smallrye:write-attribute(name=security-enabled,value=true)
```

2. Reload the server for the changes to take effect.

```
reload
```

Any subsequent attempts to access the **health** endpoint result in an authentication prompt.

25.4. ECLIPSE MICROPROFILE METRICS



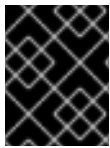
IMPORTANT

Eclipse MicroProfile Metrics is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

25.4.1. About the MicroProfile Metrics Subsystem

JBoss EAP includes the [SmallRye Metrics](#) component, which provides [Eclipse MicroProfile Metrics](#) functionality using the **microprofile-metrics-smallrye** subsystem. This subsystem is used to provide monitoring data for the JBoss EAP instance, and is enabled by default.



IMPORTANT

The **microprofile-metrics-smallrye** subsystem is only enabled in standalone configurations.

25.4.2. Examine Metrics Using the HTTP Endpoint

Metrics are automatically available on the JBoss EAP management interface, with the following contexts available.

- **/metrics/** - Contains metrics specified in the MicroProfile 3.0 specification.
- **/metrics/vendor** - Contains vendor-specific metrics, such as memory pools.
- **/metrics/application** - Contains metrics from deployed applications and subsystems that use the MicroProfile Metrics API.

The JBoss EAP subsystem metrics are exposed in the Prometheus format.

The metric names are based on subsystem and attribute names. For example, the subsystem **undertow** exposes a metric attribute **request-count** for every servlet in an application deployment. The name of this metric is **jboss_undertow_request_count**. The prefix **jboss** helps identify the metrics source.

To see a list of the metrics exposed by EAP, execute the following command in the CLI:

```
$ curl -v http://localhost:9990/metrics | grep -i type
```

Example: Obtaining the request count for a JAX-RS application

The following example demonstrates how to find the number of requests made to a web service deployed on JBoss EAP. The example uses the `helloworld-rs` quickstart as the web service. Download the quickstart from: [jboss-eap-quickstarts](#).

To examine the **request-count** metric in the `helloworld-rs` quickstart:

1. Enable statistics for the **undertow** subsystem:

- Start the standalone server with statistics enabled:

```
$ ./standalone.sh -Dwildfly.statistics-enabled=true
```

- For an already running server, enable the statistics for the **undertow** subsystem:

```
/subsystem=undertow/:write-attribute(name=statistics-enabled,value=true)
```

2. Deploy the `helloworld-rs` quickstart:

- In the root directory of the quickstart, deploy the web application using Maven:

```
$ mvn clean install wildfly:deploy
```

3. Query the **http** endpoint in the CLI using the **curl** command and filter for **request_count**:

```
$ curl -v http://localhost:9990/metrics | grep request_count
```

Output:

```
jboss_undertow_request_count_total{server="default-server",http_listener="default",} 0.0
```

The attribute value returned is **0.0**.

4. Access the quickstart, located at <http://localhost:8080/helloworld-rs/>, in a web browser and click any of the links.
5. Query the **http** endpoint again from the CLI:

```
$ curl -v http://localhost:9990/metrics | grep request_count
```

Output:

```
jboss_undertow_request_count_total{server="default-server",http_listener="default",} 1.0
```

The value is updated to **1.0**.

You can verify that the request count is updated correctly by repeating the last two steps.

25.4.3. Configure MicroProfile Metrics using the Management Console

In the management console, you can perform the following configurations:

- Enable or disable exposing metrics.
- Edit prefix for the exposed metrics.
- Enable or disable security.
- Reset non-required fields to initial or default values.

To configure the MicroProfile metrics using the management console:

- Access the management console and navigate to **Configuration → Subsystems → MicroProfile Metrics** and click **View** to open the **MicroProfile Metrics** page.

25.4.4. Enable Authentication for the HTTP Endpoint

The **metrics** context, and all subcontexts, can be configured to require that users be authorized to access the context. The following procedure enables authentication on this endpoint.

1. Set the **security-enabled** attribute to **true** on the **microprofile-metrics-smallrye** subsystem.

```
/subsystem=microprofile-metrics-smallrye:write-attribute(name=security-enabled,value=true)
```

2. Reload the server for the changes to take effect.

| reload

Any subsequent attempts to access the **metrics** endpoint will now result in an authentication prompt.

APPENDIX A. REFERENCE MATERIAL

A.1. SERVER RUNTIME ARGUMENTS

The application server startup script accepts arguments and switches at runtime. This allows the server to start under alternative configurations to those defined in the **standalone.xml**, **domain.xml**, and **host.xml** configuration files.

Alternative configurations might include starting the server with an alternative socket bindings set or a secondary configuration.

The available parameters list can be accessed by passing the help switch **-h** or **--help** at startup.

Table A.1. Runtime Switches and Arguments

Argument or Switch	Operating Mode	Description
<code>--admin-only</code>	Standalone	Set the server's running type to ADMIN_ONLY . This will cause it to open administrative interfaces and accept management requests, but not start other runtime services or accept end user requests. Note that it is recommended to use --start-mode=admin-only instead.
<code>--admin-only</code>	Domain	Set the host controller's running type to ADMIN_ONLY causing it to open administrative interfaces and accept management requests but not start servers or, if this host controller is the master for the domain, accept incoming connections from slave host controllers.
<code>-b=<value>, -b <value></code>	Standalone, Domain	Set system property jboss.bind.address , which is used in configuring the bind address for the public interface. This defaults to 127.0.0.1 if no value is specified. See the -b<interface>=<value> entry for setting the bind address for other interfaces.
<code>-b<interface>=<value></code>	Standalone, Domain	Set system property jboss.bind.address.<interface> to the given value. For example, -bmanagement=IP_ADDRESS
<code>--backup</code>	Domain	Keep a copy of the persistent domain configuration even if this host is not the domain controller.
<code>-c=<config>, -c <config></code>	Standalone	Name of the server configuration file to use. The default is standalone.xml .
<code>-c=<config>, -c <config></code>	Domain	Name of the server configuration file to use. The default is domain.xml .

Argument or Switch	Operating Mode	Description
--cached-dc	Domain	If the host is not the domain controller and cannot contact the domain controller at boot, boot using a locally cached copy of the domain configuration.
--debug [<port>]	Standalone	Activate debug mode with an optional argument to specify the port. Only works if the launch script supports it.
-D<name>[=<value>]	Standalone, Domain	Set a system property.
--domain-config=<config>	Domain	Name of the server configuration file to use. The default is domain.xml .
--git-repo	Standalone	The location of the Git repository that is used to manage and persist server configuration data. This can be local if you want to store it locally, or the URL to a remote repository.
--git-branch	Standalone	The branch or tag name in the Git repository to use. This argument should name an existing branch or tag name as it will <i>not</i> be created if it does not exist. If you use a tag name, you put the repository in a detached HEAD state, meaning future commits are not attached to any branches. Tag names are read-only and are normally used when you need to replicate a configuration across several nodes.
--git-auth	Standalone	The URL to an Elytron configuration file that contains the credentials to be used when connecting to a remote Git repository. This argument is required if your remote Git repository requires authentication. Although Git supports SSH authentication, Elytron does not; therefore, it is currently only possible to specify credentials to authenticate with HTTP or HTTPS, not SSH. This argument is not used with a local repository.
-h, --help	Standalone, Domain	Display the help message and exit.
--host-config=<config>	Domain	Name of the host configuration file to use. The default is host.xml .
--interprocess-hc-address=<address>	Domain	Address on which the host controller should listen for communication from the process controller.

Argument or Switch	Operating Mode	Description
<code>--interprocess-hc-port=<port></code>	Domain	Port on which the host controller should listen for communication from the process controller.
<code>--master-address=<address></code>	Domain	Set system property jboss.domain.master.address to the given value. In a default slave host controller configuration, this is used to configure the address of the master host controller.
<code>--master-port=<port></code>	Domain	Set system property jboss.domain.master.port to the given value. In a default slave host controller configuration, this is used to configure the port used for native management communication by the master host controller.
<code>--read-only-server-config=<config></code>	Standalone	Name of the server configuration file to use. This differs from --server-config and -c in that the original file is never overwritten.
<code>--read-only-domain-config=<config></code>	Domain	Name of the domain configuration file to use. This differs from --domain-config and -c in that the initial file is never overwritten.
<code>--read-only-host-config=<config></code>	Domain	Name of the host configuration file to use. This differs from --host-config in that the initial file is never overwritten.
<code>-P=<url>, -P <url>, --properties=<url></code>	Standalone, Domain	Load system properties from the given URL.
<code>--pc-address=<address></code>	Domain	Address on which the process controller listens for communication from processes it controls.
<code>--pc-port=<port></code>	Domain	Port on which the process controller listens for communication from processes it controls.
<code>-S<name>[=<value>]</code>	Standalone	Set a security property.
<code>-secmgr</code>	Standalone, Domain	Runs the server with a security manager installed.
<code>--server-config=<config></code>	Standalone	Name of the server configuration file to use. The default is standalone.xml .

Argument or Switch	Operating Mode	Description
<code>--start-mode=<mode></code>	Standalone	<p>Set the start mode of the server. This option cannot be used in conjunction with --admin-only. Valid values are:</p> <ul style="list-style-type: none"> ● normal: The server will start normally. ● admin-only: The server will only open administrative interfaces and accept management requests but not start other runtime services or accept end user requests. ● suspend: The server will start in suspended mode and will not service requests until it has been resumed.
<code>-u=<value>, -u <value></code>	Standalone, Domain	<p>Set system property jboss.default.multicast.address, which is used in configuring the multicast address in the socket-binding elements in the configuration files. This defaults to 230.0.0.4 if no value is specified.</p>
<code>-v, -V, --version</code>	Standalone, Domain	Display the application server version and exit.

**WARNING**

The configuration files that ship with JBoss EAP are set up to handle the behavior of the switches, for example, **-b** and **-u**. If you change your configuration files to no longer use the system property controlled by the switch, then adding it to the launch command will have no effect.

A.2. RPM SERVICE CONFIGURATION FILES

The RPM installation of JBoss EAP includes two additional configuration files compared to a ZIP or installer installation. These files are used by the service init script to specify the JBoss EAP launch environment. The location of these service configuration files differ for Red Hat Enterprise Linux 6, and Red Hat Enterprise Linux 7 and later versions.

**IMPORTANT**

For Red Hat Enterprise Linux 7 and later, RPM service configuration files are loaded using **systemd**, so variable expressions are not expanded.

Table A.2. RPM Configuration Files for Red Hat Enterprise Linux 6

File	Description
/etc/sysconfig/eap7-standalone	Settings specific to standalone JBoss EAP servers on Red Hat Enterprise Linux 6.
/etc/sysconfig/eap7-domain	Settings specific to JBoss EAP running as a managed domain on Red Hat Enterprise Linux 6.

Table A.3. RPM Configuration Files for Red Hat Enterprise Linux 7 and later

File	Description
/etc/opt/rh/eap7/wildfly/eap7-standalone.conf	Settings specific to standalone JBoss EAP servers on Red Hat Enterprise Linux 7 and later.
/etc/opt/rh/eap7/wildfly/eap7-domain.conf	Settings specific to JBoss EAP running as a managed domain on Red Hat Enterprise Linux 7 and later.

A.3. RPM SERVICE CONFIGURATION PROPERTIES

The following table shows a list of available configuration properties for the JBoss EAP RPM service along with their default values.



NOTE

If a property has the same name in both the RPM service configuration file, such as **/etc/sysconfig/eap7-standalone**, and in the JBoss EAP startup configuration file, such as **EAP_HOME/bin/standalone.conf**, the value that takes precedence is the one in the JBoss EAP startup configuration file. One such property is **JAVA_HOME**.

Table A.4. RPM Service Configuration Properties

Property	Description
JAVA_HOME	The directory where your Java Runtime Environment is installed. Default value: /usr/lib/jvm/jre
JAVAPATH	The path where the Java executable files are installed. Default value: \$JAVA_HOME/bin
WILDFLY_STARTUP_WAIT	The number of seconds that the init script will wait until confirming that the server has launched successfully after receiving a start or restart command. This property only applies to Red Hat Enterprise Linux 6. Default value: 60


Property	Description
WILDFLY_SHUTDOWN_WAIT	<p>The number of seconds that the init script will wait for the server to shutdown before continuing when it receives a stop or restart command. This property only applies to Red Hat Enterprise Linux 6.</p> <p>Default value: 20</p>
WILDFLY_CONSOLE_LOG	<p>The file that the CONSOLE log handler will be redirected to.</p> <p>Default value: /var/opt/rh/eap7/log/wildfly/standalone/console.log for a standalone server, or /var/opt/rh/eap7/log/wildfly/domain/console.log for a managed domain.</p>
WILDFLY_SH	<p>The script which is used to launch to JBoss EAP server.</p> <p>Default value: /opt/rh/eap7/root/usr/share/wildfly/bin/standalone.sh for a standalone server, or /opt/rh/eap7/root/usr/share/wildfly/bin/domain.sh for a managed domain.</p>
WILDFLY_SERVER_CONFIG	<p>The server configuration file to use.</p> <p>There is no default for this property. Either standalone.xml or domain.xml can be defined at start.</p>
WILDFLY_HOST_CONFIG	<p>For a managed domain, this property allows a user to specify the host configuration file, such as host.xml. It has no value set as the default.</p>
WILDFLY_MODULEPATH	<p>The path of the JBoss EAP module directory.</p> <p>Default value: /opt/rh/eap7/root/usr/share/wildfly/modules</p>
WILDFLY_BIND	<p>Sets the jboss.bind.address system property, which is used to configure the bind address for the public interface. This defaults to 0.0.0.0 if no value is specified.</p>
WILDFLY_OPTS	<p>Additional arguments to include on startup. For example:</p> <pre>-Dorg.wildfly.openssl.path=PATH_TO_OPENSSL_LIBS</pre>

A.4. OVERVIEW OF JBOSS EAP SUBSYSTEMS

The table below gives a brief description of the JBoss EAP subsystems.

Table A.5. JBoss EAP Subsystems

JBoss EAP Subsystem	Description
batch-jberet	Configure an environment for running batch applications and manage batch jobs.
bean-validation	Configure bean validation for validating Java object data.
core-management	Register listeners for server lifecycle events and track configuration changes .
datasources	Create and configure datasources and manage JDBC database drivers .
deployment-scanner	Configure deployment scanners to monitor particular locations for applications to deploy.
ee	Configure common functionality in the Jakarta EE platform, such as defining global modules , enabling descriptor-based property replacement , and configuring default bindings.
ejb3	Configure Enterprise JavaBeans (EJBs), including session and message-driven beans. More information for the ejb3 subsystem can be found in Developing EJB Applications for JBoss EAP.
elytron	Configure server and application security. More information on the elytron subsystem can be found in Security Architecture for JBoss EAP.
iiop-openjdk	Configure Common Object Request Broker Architecture (CORBA) services for JTS transactions and other ORB services , including security. In JBoss EAP 6, this functionality was contained in the jacorb subsystem.
infinispan	Configure caching functionality for JBoss EAP high availability services.
io	Define workers and buffer pools to be used by other subsystems.
jaxrs	Enable the deployment and functionality of JAX-RS applications.
jca	Configure the general settings for the JCA container and resource adapter deployments. The Jakarta equivalent is Jakarta Connectors.
jdr	Enable the gathering of diagnostic data to aid in troubleshooting. JBoss EAP subscribers can provide this information to Red Hat when requesting support.
jgroups	Configure the protocol stacks and communication mechanisms for how servers in a cluster talk to each other.

JBoss EAP Subsystem	Description
jmx	Configure remote Java Management Extensions (JMX) access.
jpa	<p>Manages the Jakarta Persistence 2.2 container-managed requirements and allows you to deploy persistent unit definitions, annotations, and descriptors.</p> <p>More information for the jpa subsystem can be found in the JBoss EAP Development Guide.</p>
jsf	Manage JavaServer Faces (JSF) implementations. The Jakarta equivalent is Jakarta Server Faces.
jsr77	Provide Jakarta EE management capabilities defined by the Jakarta Management specification .
logging	Configure system and application-level logging through a system of log categories and log handlers .
mail	Configure mail server attributes and custom mail transports to create a mail service that allows applications deployed to JBoss EAP to send mail using that service.
messaging-activemq	<p>Configure JMS destinations, connection factories, and other settings for Artemis, the integrated messaging provider. In JBoss EAP 6, messaging functionality was contained in the messaging subsystem.</p> <p>More information for the messaging-activemq subsystem can be found in Configuring Messaging for JBoss EAP.</p>
microprofile-config-smallrye	<p>Use MicroProfile Config SmallRye to provide portable externalization of configuration data, allowing applications to access the latest configuration properties without restarting the server.</p> <div>  <div> <p>IMPORTANT</p> <p>Eclipse Microprofile Config is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.</p> <p>See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.</p> </div> </div>

JBoss EAP Subsystem	Description
microprofile-health-smallrye	<p>Use SmallRye Health to monitor server health. See Monitor Server Health Using Eclipse MicroProfile Health for information about the microprofile-health-smallrye subsystem and how to configure it. For information about creating custom health checks, see Using Eclipse MicroProfile Health to Monitor Server Health in the <i>Development Guide</i>.</p> <div data-bbox="555 443 662 913" data-label="Image"> </div> <p>IMPORTANT</p> <p>Eclipse Microprofile Health is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.</p> <p>See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.</p>
microprofile-opentracing-smallrye	<p>Use SmallRye OpenTracing to trace requests across service boundaries. See Tracing Requests with the MicroProfile OpenTracing SmallRye Subsystem for information about the microprofile-opentracing-smallrye subsystem and how to configure it. For information about how to customize tracing for CDI beans and JAX-RS endpoints, see Using Eclipse MicroProfile OpenTracing to Trace Requests in the <i>Development Guide</i>.</p> <div data-bbox="555 1393 662 1863" data-label="Image"> </div> <p>IMPORTANT</p> <p>Eclipse Microprofile OpenTracing is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.</p> <p>See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.</p>
modcluster	Configure the server-side mod_cluster worker node .
naming	Bind entries into global JNDI namespaces and configure the remote JNDI interface.

JBoss EAP Subsystem	Description
picketlink-federation	<p>Configure PicketLink SAML-based single sign-on (SSO).</p> <p>More information on the picketlink-federation subsystem can be found in How To Set Up SSO with SAML v2 for JBoss EAP.</p>
picketlink-identity-management	Configure PicketLink identity management services. This subsystem is unsupported.
pojo	Enable deployment of applications containing JBoss Microcontainer services, as supported by previous versions of JBoss EAP.
remoting	Configure settings for inbound and outbound connections for local and remote services .
request-controller	Configure settings to suspend and shut down servers gracefully .
resource-adapters	Configure and maintain resource adapters for communication between Jakarta EE applications and an Enterprise Information System (EIS) using the Jakarta Connectors specification.
rts	Unsupported implementation of REST-AT.
sar	Enable deployment of SAR archives containing MBean services, as supported by previous versions of JBoss EAP.
security	<p>Legacy method to configure application security settings.</p> <p>More information on the security subsystem can be found in Security Architecture for JBoss EAP.</p>
security-manager	<p>Configure Java security policies to be used by the Java Security Manager.</p> <p>More information on the security-manager subsystem can be found in How to Configure Server Security for JBoss EAP.</p>
singleton	<p>Define singleton policies to configure the behavior of singleton deployments or to create singleton MSC services.</p> <p>More information on the singleton subsystem can be found in the JBoss EAP Development Guide.</p>
transactions	<p>Configure the Transaction Manager (TM) options, such as timeout values, transaction logging, and whether to use Java Transaction Service (JTS).</p> <p>More information on the transactions subsystem can be found in Managing Transactions on JBoss EAP for JBoss EAP.</p>
undertow	Configure JBoss EAP's web server and servlet container settings. In JBoss EAP 6, this functionality was contained in the web subsystem.

JBoss EAP Subsystem	Description
webservices	<p>Configure published endpoint addresses and endpoint handler chains, as well as the host name, ports, and WSDL address for the web services provider.</p> <p>More information for the webservices subsystem can be found in Developing Web Services Applications for JBoss EAP.</p>
weld	Configure Contexts and Dependency Injection (CDI) functionality for JBoss EAP.
xts	Configure settings for coordinating web services in a transaction.

A.5. ADD-USER UTILITY ARGUMENTS

The following table describes the arguments available for the **add-user.sh** or **add-user.bat** script, which is a utility for adding new users to the properties file for out-of-the-box authentication.

Table A.6. Add-User Command Arguments

Command Line Argument	Description
-a	Create a user in the application realm. If omitted, the default is to create a user in the management realm.
-dc <value>	The domain configuration directory that will contain the properties files. If it is omitted, the default directory is EAP_HOME/domain/configuration/ .
-sc <value>	An alternative standalone server configuration directory that will contain the properties files. If omitted, the default directory is EAP_HOME/standalone/configuration/ .
-up, --user-properties <value>	The name of the alternative user properties file. It can be an absolute path or it can be a file name used in conjunction with the -sc or -dc argument that specifies the alternative configuration directory.
-g, --group <value>	A comma-separated list of groups to assign to this user.
-gp, --group-properties <value>	The name of the alternative group properties file. It can be an absolute path or it can be a file name used in conjunction with the -sc or -dc argument that specifies the alternative configuration directory.
-p, --password <value>	The password of the user.

Command Line Argument	Description
<code>-u, --user <value></code>	The name of the user. User names can only contain the following characters, in any number and in any order: <ul style="list-style-type: none"> • Alphanumeric characters (a-z, A-Z, 0-9) • Dashes (-), periods (.), commas (,), at sign (@) • Backslash (\) • Equals (=)
<code>-r, --realm <value></code>	The name of the realm used to secure the management interfaces. If omitted, the default is ManagementRealm .
<code>-s, --silent</code>	Run the add-user script with no output to the console.
<code>-e, --enable</code>	Enable the user.
<code>-d, --disable</code>	Disable the user.
<code>-cw, --confirm-warning</code>	Automatically confirm warning in interactive mode.
<code>-h, --help</code>	Display usage information for the add-user script.
<code>-ds, --display-secret</code>	Print the secret value in non-interactive mode.

A.6. MANAGEMENT AUDIT LOGGING ATTRIBUTES



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-config_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.7. Management Audit Logging: Logger Attributes

Attribute	Description
<code>enabled</code>	Whether audit logging is enabled.
<code>log-boot</code>	Whether operations should be logged on server boot.
<code>log-read-only</code>	Whether operations that do not modify the configuration or any runtime services should be logged.

Table A.8. Management Audit Logging: Log Formatter Attributes

Attribute	Description
compact	If true , it will format the JSON on one line. There may still be values containing new lines, so if having the whole record on one line is important, set escape-new-line or escape-control-characters to true .
date-format	The date format to use as understood by java.text.SimpleDateFormat . This is ignored if include-date is set to false .
date-separator	The separator between the date and the rest of the formatted log message. This is ignored if include-date is set to false .
escape-control-characters	If true , it will escape all control characters, ASCII entries with a decimal value greater than 32 , with the ASCII code in octal. For example, a new line becomes #012 . If true , this will override escape-new-line=false .
escape-new-line	If true , it will escape all new lines with the ASCII code in octal #012 .
include-date	Whether or not to include the date in the formatted log record.

Table A.9. Management Audit Logging: File Handler Attributes

Attribute	Description
disabled-due-to-failure	Whether this handler has been disabled due to logging failures (read-only).
failure-count	The number of logging failures since the handler was initialized (read-only).
formatter	The JSON formatter used to format the log messages.
max-failure-count	The maximum number of logging failures before disabling this handler.
path	The path of the audit log file.
relative-to	The name of another previously named path, or of one of the standard paths provided by the system. If relative-to is provided, the value of the path attribute is treated as relative to the path specified by this attribute.
rotate-at-startup	Whether the old log file should be rotated at server startup.

Table A.10. Management Audit Logging: Syslog Handler Attributes

Attribute	Description
app-name	The application name to add to the syslog records as defined in section 6.2.5 of <i>RFC-5424</i> . If not specified it will default to the name of the product.
disabled-due-to-failure	Whether this handler has been disabled due to logging failures (read-only).
facility	The facility to use for syslog logging as defined in section 6.2.1 of <i>RFC-5424</i> and section 4.1.1 of <i>RFC-3164</i> .
failure-count	The number of logging failures since the handler was initialized (read-only).
formatter	The JSON formatter used to format the log messages.
max-failure-count	The maximum number of logging failures before disabling this handler.
max-length	The maximum length in bytes a log message, including the header, is allowed to be. If undefined, it will default to 1024 bytes if the syslog-format is RFC3164 , or 2048 bytes if the syslog-format is RFC5424 .
protocol	The protocol to use for the syslog handler. Must be one and only one of udp , tcp or tls .
syslog-format	The syslog format: RFC5424 or RFC3164 .
truncate	Whether or not a message, including the header, should truncate the message if the length in bytes is greater than the value of the max-length attribute. If set to false , messages will be split and sent with the same header values.

**NOTE**

Syslog servers vary in their implementation, so not all settings are applicable to all syslog servers. Testing has been conducted using the *rsyslog* syslog implementation.

This table lists only the high-level attributes. Each attribute has configuration parameters, and some have child configuration parameters.

A.7. INTERFACE ATTRIBUTES

**NOTE**

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-config_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.11. Interface Attributes and Values

Interface Element	Description
any	Element indicating that part of the selection criteria for an interface should be that it meets at least one, but not necessarily all, of the nested set of criteria.
any-address	Empty element indicating that sockets using this interface should be bound to a wildcard address. The IPv6 wildcard address (::) will be used unless the java.net.preferIPv4Stack system property is set to true, in which case the IPv4 wildcard address (0.0.0.0) will be used. If a socket is bound to an IPv6 anylocal address on a dual-stack machine, it can accept both IPv6 and IPv4 traffic; if it is bound to an IPv4 (IPv4-mapped) anylocal address, it can only accept IPv4 traffic.
inet-address	Either an IP address in IPv6 or IPv4 dotted decimal notation, or a host name that can be resolved to an IP address.
link-local-address	Empty element indicating that part of the selection criteria for an interface should be whether or not an address associated with it is link-local.
loopback	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a loopback interface.
loopback-address	A loopback address that may not actually be configured on the machine's loopback interface. Differs from inet-address type in that the given value will be used even if no NIC can be found that has the IP address associated with it.
multicast	Empty element indicating that part of the selection criteria for an interface should be whether or not it supports multicast.
name	The name of the interface.
nic	The name of a network interface (e.g. eth0, eth1, lo).
nic-match	A regular expression against which the names of the network interfaces available on the machine can be matched to find an acceptable interface.
not	Element indicating that part of the selection criteria for an interface should be that it does not meet any of the nested set of criteria.
point-to-point	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a point-to-point interface.
public-address	Empty element indicating that part of the selection criteria for an interface should be whether or not it has a publicly routable address.

Interface Element	Description
site-local-address	Empty element indicating that part of the selection criteria for an interface should be whether or not an address associated with it is site-local.
subnet-match	A network IP address and the number of bits in the address' network prefix, written in <i>slash notation</i> , for example, 192.168.0.0/16 .
up	Empty element indicating that part of the selection criteria for an interface should be whether or not it is currently up.
virtual	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a virtual interface.

A.8. SOCKET BINDING ATTRIBUTES



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-config_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

The following tables show the attributes that can be configured for each of the three types of socket bindings.

- [socket-binding](#)
- [remote-destination-outbound-socket-binding](#)
- [local-destination-outbound-socket-binding](#)

Table A.12. Inbound Socket Binding ([socket-binding](#)) Attributes

Attribute	Description
client-mappings	Specifies the client mappings for this socket binding. A client connecting to this socket should use the destination address specified in the mapping that matches its desired outbound interface. This allows for advanced network topologies that use either network address translation, or have bindings on multiple network interfaces to function. Each mapping should be evaluated in declared order, with the first successful match used to determine the destination.
fixed-port	Whether the port value should remain fixed even if numeric offsets are applied to the other sockets in the socket group.

Attribute	Description
interface	Name of the interface to which the socket should be bound, or, for multicast sockets, the interface on which it should listen. This should be one of the declared interfaces. If not defined, the value of the default-interface attribute from the enclosing socket binding group will be used.
multicast-address	Multicast address on which the socket should receive multicast traffic. If unspecified, the socket will not be configured to receive multicast.
multicast-port	Port on which the socket should receive multicast traffic. Must be configured if multicast-address is configured.
name	The name of the socket. Services needing to access the socket configuration information will find it using this name. This attribute is required.
port	Number of the port to which the socket should be bound. Note that this value can be overridden if servers apply a port-offset to increment or decrement all port values.

Table A.13. Remote Outbound Socket Binding (remote-destination-outbound-socket-binding**) Attributes**

Attribute	Description
fixed-source-port	Whether the port value should remain fixed even if numeric offsets are applied to the other outbound sockets in the socket group.
host	The host name or IP address of the remote destination to which this outbound socket will connect.
port	The port number of the remote destination to which the outbound socket should connect.
source-interface	The name of the interface that will be used for the source address of the outbound socket.
source-port	The port number that will be used as the source port of the outbound socket.

Table A.14. Local Outbound Socket Binding (local-destination-outbound-socket-binding**) Attributes**

Attribute	Description
fixed-source-port	Whether the port value should remain fixed even if numeric offsets are applied to the other outbound sockets in the socket group.

Attribute	Description
socket-binding-ref	The name of the local socket binding that will be used to determine the port to which this outbound socket connects.
source-interface	The name of the interface that will be used for the source address of the outbound socket.
source-port	The port number that will be used as the source port of the outbound socket.

A.9. DEFAULT SOCKET BINDINGS

The following tables show the default socket bindings for each socket binding group.

- [standard-sockets](#)
- [ha-sockets](#)
- [full-sockets](#)
- [full-ha-sockets](#)
- [load-balancer-sockets](#)

Table A.15. standard-sockets

Socket Binding	Port	Description
ajp	8009	Apache JServ Protocol. Used for HTTP clustering and load balancing.
http	8080	The default port for deployed web applications.
https	8443	SSL-encrypted connection between deployed web applications and clients.
management-http	9990	Used for HTTP communication with the management layer.
management-https	9993	Used for HTTPS communication with the management layer.
txn-recovery-environment	4712	The JTA transaction recovery manager.
txn-status-manager	4713	The JTA / JTS transaction manager.

Table A.16. ha-sockets

Socket Binding	Port	Multicast Port	Description
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.
http	8080		The default port for deployed web applications.
https	8443		SSL-encrypted connection between deployed web applications and clients.
jgroups-mping		45700	Multicast. Used to discover initial membership in a HA cluster.
jgroups-tcp	7600		Unicast peer discovery in HA clusters using TCP.
jgroups-udp	55200	45688	Multicast peer discovery in HA clusters using UDP.
management-http	9990		Used for HTTP communication with the management layer.
management-https	9993		Used for HTTPS communication with the management layer.
modcluster		23364	Multicast port for communication between JBoss EAP and the HTTP load balancer.
txn-recovery-environment	4712		The JTA transaction recovery manager.
txn-status-manager	4713		The JTA / JTS transaction manager.

Table A.17. full-sockets

Socket Binding	Port	Description
ajp	8009	Apache JServ Protocol. Used for HTTP clustering and load balancing.
http	8080	The default port for deployed web applications.
https	8443	SSL-encrypted connection between deployed web applications and clients.

Socket Binding	Port	Description
iiop	3528	CORBA services for JTS transactions and other ORB-dependent services.
iiop-ssl	3529	SSL-encrypted CORBA services.
management-http	9990	Used for HTTP communication with the management layer.
management-https	9993	Used for HTTPS communication with the management layer.
txn-recovery-environment	4712	The JTA transaction recovery manager.
txn-status-manager	4713	The JTA / JTS transaction manager.

Table A.18. full-ha-sockets

Name	Port	Multicast Port	Description
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.
http	8080		The default port for deployed web applications.
https	8443		SSL-encrypted connection between deployed web applications and clients.
iiop	3528		CORBA services for JTS transactions and other ORB-dependent services.
iiop-ssl	3529		SSL-encrypted CORBA services.
jgroups-mping		45700	Multicast. Used to discover initial membership in a HA cluster.
jgroups-tcp	7600		Unicast peer discovery in HA clusters using TCP.
jgroups-udp	55200	45688	Multicast peer discovery in HA clusters using UDP.
management-http	9990		Used for HTTP communication with the management layer.

Name	Port	Multicast Port	Description
management-https	9993		Used for HTTPS communication with the management layer.
modcluster		23364	Multicast port for communication between JBoss EAP and the HTTP load balancer.
txn-recovery-environment	4712		The JTA transaction recovery manager.
txn-status-manager	4713		The JTA / JTS transaction manager.

Table A.19. load-balancer-sockets

Name	Port	Multicast Port	Description
http	8080		The default port for deployed web applications.
https	8443		SSL-encrypted connection between deployed web applications and clients.
management-http	9990		Used for HTTP communication with the management layer.
management-https	9993		Used for HTTPS communication with the management layer.
mcmp-management	8090		The port for the Mod-Cluster Management Protocol (MCMP) connection to transmit lifecycle events.
modcluster		23364	Multicast port for communication between JBoss EAP and the HTTP load balancer.

A.10. MODULE COMMAND ARGUMENTS

The following arguments can be passed to the **module add** management CLI command:

Table A.20. Module Command Arguments

Argument	Description
<code>--absolute-resources</code>	<p>Use this argument to specify a list of absolute file system paths to reference from its module.xml file. The files specified are not copied to the module directory.</p> <p>See --resource-delimiter for delimiter details.</p>
<code>--allow-nonexistent-resources</code>	<p>Use this argument to create empty directories for resources specified by --resources that do not exist. The module add command will fail if there are resources that do not exist and this argument is not used.</p>
<code>--dependencies</code>	<p>Use this argument to provide a comma-separated list of module names that this module depends on.</p>
<code>--export-dependencies</code>	<p>Use this argument to specify exported dependencies.</p> <pre>module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --export-dependencies=javax.api,javax.transaction.api</pre>
<code>--main-class</code>	<p>Use this argument to specify the fully qualified class name that declares the module's main method.</p>
<code>--module-root-dir</code>	<p>Use this argument if you have defined an external JBoss EAP module directory to use instead of the default EAP_HOME/modules/ directory.</p> <pre>module add --module-root-dir=/path/to/my-external-modules/ --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --dependencies=javax.api,javax.transaction.api</pre>
<code>--module-xml</code>	<p>Use this argument to provide a file system path to a module.xml to use for this new module. This file is copied to the module directory. If this argument is not specified, a module.xml file is generated in the module directory.</p>
<code>--name</code>	<p>Use this argument to provide the name of the module to add. This argument is required.</p>
<code>--properties</code>	<p>Use this argument to provide a comma-separated list of PROPERTY_NAME=PROPERTY_VALUE pairs that define module properties.</p>
<code>--resource-delimiter</code>	<p>Use this argument to set a user-defined file path separator for the list of resources provided to the --resources or absolute-resources argument. If not set, the file path separator is a colon (:) for Linux and a semicolon (;) for Windows.</p>

Argument	Description
<code>--resources</code>	<p>Use this argument to specify the resources for this module by providing a list of file system paths. The files are copied to this module directory and referenced from its module.xml file. If you provide a path to a directory, the directory and its contents are copied to the module directory. Symbolic links are not preserved; linked resources are copied to the module directory. This argument is required unless --absolute-resources or --module-xml is provided.</p> <p>See --resource-delimiter for delimiter details.</p>
<code>--slot</code>	<p>Use this argument to add the module to a slot other than the default main slot.</p> <pre>module add --name=com.mysql --slot=8.0 -- resources=/path/to/mysql-connector-java-8.0.12.jar -- dependencies=javax.api,javax.transaction.api</pre>

A.11. DEPLOYMENT SCANNER MARKER FILES

Marker files are used by the deployment scanner to mark the status of an application within the deployment directory of the JBoss EAP server instance. A marker file has the same name as the deployment, with the file suffix indicating the state of the application's deployment.

For example, a successful deployment of **test-application.war** would have a marker file named **test-application.war.deployed**.

The following table lists the available marker file types and their meanings.

Table A.21. Marker File Types

Filename Suffix	Origin	Description
<code>.deployed</code>	System-generated	Indicates that the content has been deployed. The content will be undeployed if this file is deleted.
<code>.dodeploy</code>	User-generated	Indicates that the content should be deployed or redeployed.
<code>.failed</code>	System-generated	Indicates deployment failure. The marker file contains information about the cause of failure. If the marker file is deleted, the content will be eligible for auto-deployment again.
<code>.isdeploying</code>	System-generated	Indicates that the deployment is in progress. This marker file will be deleted upon completion.

Filename Suffix	Origin	Description
.isundeploying	System-generated	Triggered by deleting a .deployed file, this indicates that the content is being undeployed. This marker file will be deleted upon completion.
.pending	System-generated	Indicates that the deployment scanner recognizes the need to deploy content, but an issue is currently preventing auto-deployment (for example, if content is in the process of being copied). This marker serves as a global deployment road-block, meaning that the scanner will not instruct the server to deploy or undeploy <i>any</i> content while this marker file exists.
.skipdeploy	User-generated	Disables auto-deploy of an application while present. Useful as a method of temporarily blocking the auto-deployment of exploded content, preventing the risk of incomplete content edits being pushed. Can be used with zipped content, although the scanner detects in-progress changes to zipped content and waits until completion.
.undeployed	System-generated	Indicates that the content has been undeployed. Deletion of this marker file has no impact to content redeployment.

A.12. DEPLOYMENT SCANNER ATTRIBUTES

The deployment scanner contains the following configurable attributes.



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/jboss-as-deployment-scanner_2_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

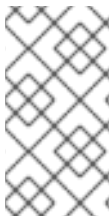
Table A.22. Deployment Scanner Attributes

Name	Default	Description
auto-deploy-exploded	false	Allows the automatic deployment of exploded content without requiring a .dodeploy marker file. Recommended for only basic development scenarios to prevent exploded application deployment from occurring during changes by the developer or operating system.
auto-deploy-xml	true	Allows the automatic deployment of XML content without requiring a .dodeploy marker file.

Name	Default	Description
auto-deploy-zipped	true	Allows the automatic deployment of zipped content without requiring a .dodeploy marker file.
deployment-timeout	600	The time value in seconds for the deployment scanner to allow a deployment attempt before being canceled.
path	deployments	The actual file system path to be scanned. Treated as an absolute path, unless the relative-to attribute is specified, in which case the value is treated as relative to that path.
relative-to	jboss.server.base.dir	Reference to a file system path defined as a path in the server configuration.
runtime-failure-causes-rollback	false	Whether a runtime failure of a deployment causes a rollback of the deployment as well as all other (possibly unrelated) deployments as part of the scan operation.
scan-enabled	true	Allows the automatic scanning for applications by scan-interval and at startup.
scan-interval	5000	The time interval in milliseconds that the repository should be scanned for changes. A value of less than 1 causes the scan to occur only at initial startup.

A.13. MANAGED DOMAIN JVM CONFIGURATION ATTRIBUTES

The following JVM configuration options can be set for a managed domain at the host, server group, or server level. Note that valid values for some of these attributes are dependent upon your JVM. See your JDK vendor's documentation for additional information.



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-config_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.23. JVM Configuration Attributes

Attribute	Description
agent-lib	Sets the value of the -agentlib java option, which specifies the Java agent library.

Attribute	Description
agent-path	Sets the value of the -agentpath java option, which specifies the Java agent path.
debug-enabled	Whether to enable debug. This attribute only applies to JVM configurations at the server level.
debug-options	Specifies the JVM options to use when debug is enabled. This attribute only applies to JVM configurations at the server level.
env-classpath-ignored	Whether to ignore the CLASSPATH environment variable.
environment-variables	Specifies key/value pair environment variables.
heap-size	Sets the value of the -Xms option, which specifies the initial heap size allocated by the JVM.
java-agent	Sets the value of the -javaagent java option, which specifies the Java agent.
java-home	Sets the value of the JAVA_HOME variable.
jvm-options	Specifies any additional JVM options needed.
launch-command	Specifies an operating system level command to prefix before the java command used to launch the server process. For example, you could use the sudo command to run the Java process as another user.
max-heap-size	Sets the value of the -Xmx option, which specifies the maximum heap size allocated by the JVM.
max-permgen-size	Sets the maximum size of the permanent generation. <i>Deprecated: The JVM no longer provides a separate permanent generation space.</i>
permgen-size	Sets the initial permanent generation size. <i>Deprecated: The JVM no longer provides a separate permanent generation space.</i>
stack-size	Sets the value of the -Xss option, which specifies the JVM stack size.
type	Specifies which vendor provided the JVM in use. Available options are ORACLE , IBM , SUN , or OTHER .

A.14. MAIL SUBSYSTEM ATTRIBUTES

The following tables describe the attributes in the **mail** subsystem for [mail sessions](#) and the following mail server types:

- [imap](#)
- [pop3](#)
- [smtp](#)
- [custom](#)



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-mail_3_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.24. Mail Session Attributes

Attribute	Description
debug	Whether to enable Jakarta Mail debugging.
from	The default "from" address to use if not set when sending.
jndi-name	The JNDI name to which the mail session should be bound.

Table A.25. IMAP Mail Server Attributes

Attribute	Description
credential-reference	Credential, from a credential store, to authenticate on the server.
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

Table A.26. POP3 Mail Server Attributes

Attribute	Description
credential-reference	Credential, from a credential store, to authenticate on the server.

Attribute	Description
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

Table A.27. SMTP Mail Server Attributes

Attribute	Description
credential-reference	Credential, from a credential store to authenticate on the server.
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

Table A.28. Custom Mail Server Attributes

Attribute	Description
credential-reference	Credential, from a credential store, to authenticate on the server.
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
properties	The Jakarta Mail properties for this server.
ssl	Whether the server requires SSL.

Attribute	Description
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

A.15. ROOT LOGGER ATTRIBUTES



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/jboss-as-logging_3_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.29. Root Logger Attributes

Attribute	Description
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that excludes log entries that do <i>not</i> match a pattern: not(match("WFLY.*"))
handlers	A list of log handlers that are used by the root logger.
level	The lowest level of log message that the root logger records.



NOTE

A **filter-spec** specified for the root logger is *not* inherited by other handlers. Instead a **filter-spec** must be specified per handler.

A.16. LOG CATEGORY ATTRIBUTES



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/jboss-as-logging_6_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.30. Log Category Attributes

Attribute	Description
category	The log category from which log messages will be captured.

Attribute	Description
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
handlers	A list of log handlers associated with the logger.
level	The lowest level of log message that the log category records.
use-parent-handlers	If set to true , this category will use the log handlers of the root logger in addition to any other assigned handlers.

A.17. LOG HANDLER ATTRIBUTES



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/jboss-as-logging_6_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.31. Console Log Handler Attributes

Attribute	Description
autoflush	If set to true , the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>

Attribute	Description
named-formatter	The name of the defined formatter to be used on the handler.
target	<p>The system output stream where the output of the log handler is sent. This can be one of the following:</p> <ul style="list-style-type: none"> • System.err: Log handler output goes to the system error stream. • System.out: Log handler output goes to the standard output stream. • console: Log handler output goes to the java.io.PrintWriter class.

Table A.32. File Log Handler Attributes

Attribute	Description
append	If set to true , all messages written by this handler will be appended to the file if it already exists. If set to false , a new file will be created each time the application server launches.
autoflush	If set to true , the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	The object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>
named-formatter	The name of the defined formatter to be used on the handler.

Table A.33. Periodic Log Handler Attributes

Attribute	Description
append	If set to true , all messages written by this handler will be appended to the file if it already exists. If set to false , a new file will be created each time the application server launches.
autoflush	If set to true , the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	Object that represents the file to which the output of this log handler is written. It has two configuration properties, relative-to and path .
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*")) .
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>
named-formatter	The name of the defined formatter to be used on the handler.
suffix	This string is included in the suffix appended to rotated logs. The format of the suffix is a dot (.) followed by a date string which is able to be parsed by the SimpleDateFormat class.

Table A.34. Size Log Handler Attributes

Attribute	Description
append	If set to true , all messages written by this handler will be appended to the file if it already exists. If set to false , a new file will be created each time the application server launches.
autoflush	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt.

Attribute	Description
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	Object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec</i> .
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
max-backup-index	<p>The maximum number of rotated logs that are kept. When this number is reached, the oldest log is reused. The default is 1.</p> <p>If the suffix attribute is used, the suffix of rotated log files is included in the rotation algorithm. When the log file is rotated, the oldest file whose name starts with name+suffix is deleted, the remaining rotated log files have their numeric suffix incremented and the newly rotated log file is given the numeric suffix 1.</p>
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>
named-formatter	The name of the defined formatter to be used on the handler.
rotate-on-boot	If set to true , a new log file will be created on server restart. The default is false .
rotate-size	The maximum size that the log file can reach before it is rotated. A single character appended to the number indicates the size units: b for bytes, k for kilobytes, m for megabytes, g for gigabytes. For example, 50m for 50 megabytes.
suffix	This string is included in the suffix appended to rotated logs. The format of the suffix is a dot (.) followed by a date string which is able to be parsed by the SimpleDateFormat class.

Table A.35. Periodic Size Log Handler Attributes

Attribute	Description
append	If set to true , all messages written by this handler will be appended to the file if it already exists. If set to false , a new file will be created each time the application server launches.

Attribute	Description
autoflush	If set to true , the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	Object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
max-backup-index	<p>The maximum number of rotated logs that are kept. When this number is reached, the oldest log is reused. The default is 1.</p> <p>If the suffix attribute is used, the suffix of rotated log files is included in the rotation algorithm. When the log file is rotated, the oldest file whose name starts with name+suffix is deleted, the remaining rotated log files have their numeric suffix incremented and the newly rotated log file is given the numeric suffix 1.</p>
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>
named-formatter	The name of the defined formatter to be used on the handler.
rotate-on-boot	If set to true , a new log file will be created on server restart. The default is false .
rotate-size	The maximum size that the log file can reach before it is rotated. A single character appended to the number indicates the size units: b for bytes, k for kilobytes, m for megabytes, g for gigabytes. For example, 50m for 50 megabytes.
suffix	This string is included in the suffix appended to rotated logs. The format of the suffix is a dot (.) followed by a date string which is able to be parsed by the SimpleDateFormat class.

Table A.36. Syslog Handler Attributes

Attribute	Description
app-name	The app name used when formatting the message in RFC5424 format. By default the app name is java .

Attribute	Description
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
facility	The facility as defined by RFC-5424 and RFC-3164.
hostname	The name of the host from which the messages are being sent. For example, the name of the host the application server is running on.
level	The lowest level of log message the log handler records.
port	The port on which the syslog server is listening.
server-address	The address of the syslog server.
syslog-format	Formats the log message according to the RFC specification.
named-formatter	Formats the message of the syslog payload. With this attribute, you can customize the message as required.

Table A.37. Socket Log Handler Attributes

Attribute	Description
autoflush	Whether to automatically flush after each write.
block-on-reconnect	If set to true , the write methods will block when attempting to reconnect. This is only advisable to be set to true if using an asynchronous handler.
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding used by this handler
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
level	The lowest level of log message the log handler records.
named-formatter	The name of the defined formatter to be used on the handler.
outbound-socket-binding-ref	The reference to the outbound socket binding for the socket connection.
protocol	The protocol the socket should communicate over. Allowed values are TCP , UDP , or SSL_TCP .

Attribute	Description
ssl-context	The reference to the defined SSL context. This is only used if protocol is set to SSL_TCP .

Table A.38. Custom Log Handler Attributes

Attribute	Description
class	The logging handler class to be used.
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
module	The module one which the logging handler depends.
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>
named-formatter	The name of the defined formatter to be used on the handler.
properties	The properties used for the logging handler.

Table A.39. Async Log Handler Attributes

Attribute	Description
enabled	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter	Defines a simple filter type. <i>Deprecated in favor of filter-spec.</i>
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("WFLY.*"))

Attribute	Description
level	The lowest level of log message the log handler records.
name	The name of the log handler. <i>Deprecated since the handler's address contains the name.</i>
overflow-action	How this handler responds when its queue length is exceeded. This can be set to BLOCK or DISCARD . BLOCK makes the logging application wait until there is available space in the queue. This is the same behavior as an non-async log handler. DISCARD allows the logging application to continue but the log message is deleted.
queue-length	Maximum number of log messages that will be held by this handler while waiting for sub-handlers to respond.
subhandlers	The list of log handlers to which this async handler passes its log messages.

A.18. LOG FORMATTER ATTRIBUTES

Table A.40. Format Characters for Pattern Formatter

Symbol	Description
%c	The category of the logging event.
%p	The level of the log entry (INFO, DEBUG, etc.).
%P	The localized level of the log entry.
%d	The current date/time (yyyy-MM-dd HH:mm:ss,SSS format).
%r	The relative time (milliseconds since the log was initialized).
%z	The time zone, which must be specified before the date (%d). For example, %z{GMT}%d{HH:mm:ss,SSS} .
%k	A log resource key (used for localization of log messages).
%m	The log message (including exception trace).
%s	The simple log message (no exception trace).
%e	The exception stack trace (no extended module information).
%E	The exception stack trace (with extended module information).

Symbol	Description
%t	The name of the current thread.
%n	A newline character.
%C	The class of the code calling the log method (slow).
%F	The filename of the class calling the log method (slow).
%l	The source location of the code calling the log method (slow).
%L	The line number of the code calling the log method (slow).
%M	The method of the code calling the log method (slow).
%x	The Nested Diagnostic Context.
%X	The Message Diagnostic Context.
%%	A literal percent (%) character (escaping).

Table A.41. JSON Log Formatter Attributes

Attribute	Description
date-format	The date-time format pattern. The pattern must be a valid java.time.format.DateTimeFormatter.ofPattern() pattern. The default pattern is an ISO-8601 extended offset date-time format.
exception-output-type	Indicates how the cause of the logged message, if one is available, is added to the JSON output. The allowed values are: <ul style="list-style-type: none"> ● detailed ● formatted ● detailed-and-formatted
key-overrides	Allows the names of the keys for the JSON properties to be overridden.
meta-data	Sets the metadata to be used in the JSON formatter.
pretty-print	Whether or not pretty printing should be used when formatting.



Attribute	Description
print-details	<p>Whether or not details should be printed. The details include the source class name, source file name, source method name, source module name, source module version and source line number.</p> <div>  <div> <p>NOTE</p> <p>Printing the details can be expensive as the values are retrieved from the caller.</p> </div> </div>
record-delimiter	The value to be used to indicate the end of a record. If set to null no delimiter will be used at the end of the record. The default value is a line feed.
zone-id	The zone ID for formatting the date and time. The system default is used if left undefined.

Table A.42. XML Log Formatter Attributes

Attribute	Description
date-format	The date-time format pattern. The pattern must be a valid java.time.format.DateTimeFormatter.ofPattern() pattern. The default pattern is an ISO-8601 extended offset date-time format.
exception-output-type	<p>Indicates how the cause of the logged message, if one is available, is added to the XML output. The allowed values are:</p> <ul style="list-style-type: none"> • detailed • formatted • detailed-and-formatted
key-overrides	Allows the names of the keys for the XML properties to be overridden.
meta-data	Sets the meta data to use in the XML format. Properties are added to each log message.
namespace-uri	Sets the namespace URI used for each record if print-namespace attribute is true. Note that if no namespace-uri is defined and there are overridden keys no namespace will be written regardless if the print-namespace attribute is set to true.
pretty-print	Whether or not pretty printing should be used when formatting.

Attribute	Description
print-details	<p>Whether or not details should be printed. The details include the source class name, source file name, source method name, source module name, source module version and source line number.</p> <div>  <div> NOTE <p>Printing the details can be expensive as the values are retrieved from the caller.</p> </div> </div>
record-delimiter	The value to be used to indicate the end of a record. If this is null, no delimiter is used at the end of the record. The default value is a line feed.
zone-id	The zone ID for formatting the date and time. The system default is used if left undefined.

A.19. DATASOURCE CONNECTION URLS

Table A.43. Datasource Connection URLs

Datasource	Connection URL
IBM DB2	<code>jdbc:db2://SERVER_NAME:PORT/DATABASE_NAME</code>
MariaDB	<code>jdbc:mariadb://SERVER_NAME:PORT/DATABASE_NAME</code>
MariaDB Galera Cluster	<code>jdbc:mariadb://SERVER_NAME:PORT,SERVER_NAME:PORT/DATABASE_NAME</code>
Microsoft SQL Server	<code>jdbc:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME</code>
MySQL	<code>jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME</code>
Oracle	<code>jdbc:oracle:thin:@SERVER_NAME:PORT:ORACLE_SID</code>
PostgreSQL	<code>jdbc:postgresql://SERVER_NAME:PORT/DATABASE_NAME</code>
Sybase	<code>jdbc:sybase:Tds:SERVER_NAME:PORT/DATABASE_NAME</code>

A.20. DATASOURCE ATTRIBUTES



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-datasources_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.44. Datasource Attributes

Attribute	Datasource Type	Description
allocation-retry	Non-XA, XA	The number of times that allocating a connection should be tried before throwing an exception. The default is 0 , so an exception is thrown upon the first failure.
allocation-retry-wait-millis	Non-XA, XA	The amount of time, in milliseconds, to wait between retrying to allocate a connection. The default is 0 ms.
allow-multiple-users	Non-XA, XA	Whether multiple users will access the datasource through the getConnection(user, password) method and if the internal pool type accounts for this behavior.
authentication-context	Non-XA, XA	The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.
background-validation	Non-XA, XA	Whether connections should be validated on a background thread versus being validated prior to use. Background validation is typically not to be used with validate-on-match or there will be redundant checks. With background validation, there is an opportunity for a connection to go bad between the time of the validations can and being handed to the client, so the application must account for this possibility.
background-validation-millis	Non-XA, XA	The frequency, in milliseconds, that background validation will run.
blocking-timeout-wait-millis	Non-XA, XA	The maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time.
capacity-decrementer-class	Non-XA, XA	Class defining the policy for decrementing connections in the pool.

Attribute	Datasource Type	Description
capacity-decrementer-properties	Non-XA, XA	Properties to be injected in the class defining the policy for decrementing connections in the pool.
capacity-incrementer-class	Non-XA, XA	Class defining the policy for incrementing connections in the pool.
capacity-incrementer-properties	Non-XA, XA	Properties to be injected in the class defining the policy for incrementing connections in the pool.
check-valid-connection-sql	Non-XA, XA	An SQL statement to check validity of a pool connection. This may be called when a managed connection is obtained from the pool.
connectable	Non-XA, XA	Enable the use of CMR, which means that a local resource can reliably participate in an XA transaction.
connection-listener-class	Non-XA, XA	Specifies class name extending org.jboss.jca.adapters.jdbc.spi.listener.ConnectionListener . This class listens for connection activation and passivation in order to perform actions before the connection is returned to the application or to the pool. The specified class must be bundled together with the JDBC driver in one module using two resource JARs, as seen in Installing a JDBC Driver as a Core Module , or in a separate global module, as seen in Define Global Modules .
connection-listener-property	Non-XA, XA	Properties to be injected into the class specified in the connection-listener-class . The properties injected are compliant with the JavaBeans conventions. For example, if you specify a property named foo , then the connection listener class needs to have a method setFoo that accepts String as argument.
connection-properties	Non-XA Only	Arbitrary string name/value pair connection properties to pass to the Driver.connect(url, props) method.
connection-url	Non-XA Only	The JDBC driver connection URL.
credential-reference	Non-XA, XA	Credential, from a credential store, to authenticate on datasource.
datasource-class	Non-XA Only	The fully-qualified name of the JDBC datasource class.

Attribute	Datasource Type	Description
driver-class	Non-XA Only	The fully-qualified name of the JDBC driver class.
driver-name	Non-XA, XA	Defines the JDBC driver the datasource should use. It is a symbolic name matching the name of installed driver. If the driver is deployed as JAR, the name is the name of the deployment.
elytron-enabled	Non-XA, XA	Enables Elytron security for handling authentication of connections. The Elytron authentication-context to be used will be current context if no context is specified. See authentication-context for additional information.
enabled	Non-XA, XA	Whether the datasource should be enabled.
enlistment-trace	Non-XA, XA	Whether enlistment traces should be recorded. This is false by default.
exception-sorter-class-name	Non-XA, XA	An instance of org.jboss.jca.adapters.jdbc.ExceptionSorter that provides a method to validate if an exception should broadcast an error.
exception-sorter-properties	Non-XA, XA	The exception sorter properties.

Attribute	Datasource Type	Description
flush-strategy	Non-XA, XA	<p>Specifies how the pool should be flushed in case of an error. Valid values are:</p> <p>FailingConnectionOnly Only the failing connection is removed. This is the default setting.</p> <p>InvalidIdleConnections The failing connection and idle connections that share the same credentials and are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed.</p> <p>IdleConnections The failing connection and idle connections that share the same credentials are removed.</p> <p>Gracefully The failing connection and idle connections that share the same credentials are removed. Active connections that share the same credentials are destroyed upon return to the pool.</p> <p>EntirePool The failing connection and idle and active connections that share the same credentials are removed. This setting is not recommended for production systems.</p> <p>AllInvalidIdleConnections The failing connection and idle connections that are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed.</p> <p>AllIdleConnections The failing connection and all idle connections are removed.</p> <p>AllGracefully The failing connection and all idle connections are removed. Active connections are destroyed upon return to the pool.</p> <p>AllConnections The failing connection and all idle and active connections are removed. This setting is not recommended for production systems.</p>
idle-timeout-minutes	Non-XA, XA	<p>The maximum time, in minutes, a connection may be idle before being closed. If not specified, the default is 30 minutes. The actual maximum time also depends on the IdleRemover scan time, which is half of the smallest idle-timeout-minutes value of any pool.</p>
initial-pool-size	Non-XA, XA	<p>The initial number of connections a pool should hold.</p>

Attribute	Datasource Type	Description
interleaving	XA Only	Whether to enable interleaving for XA connections.
jndi-name	Non-XA, XA	The unique JNDI name for the datasource.
jta	Non-XA Only	Enable JTA integration.
max-pool-size	Non-XA, XA	The maximum number of connections that a pool can hold.
mcp	Non-XA, XA	The ManagedConnectionPool implementation. For example, org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool
min-pool-size	Non-XA, XA	The minimum number of connections that a pool can hold.
new-connection-sql	Non-XA, XA	An SQL statement to execute whenever a connection is added to the connection pool.
no-recovery	XA Only	Whether the connection pool should be excluded from recovery.
no-tx-separate-pool	XA Only	Whether to create a separate sub-pool for each context. This may be required for some Oracle datasources, which may not allow XA connections to be used both inside and outside of a JTA transaction. Using this option will cause your total pool size to be twice the max-pool-size , because two actual pools will be created.
pad-xid	XA Only	Whether to pad the Xid.
password	Non-XA, XA	The password to use when creating a new connection.
pool-fair	Non-XA, XA	Defines if pool should be fair. This setting is part of a Semaphore class used to manage the connection pools in Jakarta Connectors, which provides a performance benefit in some use cases where the order of leasing connections is not required.
pool-prefill	Non-XA, XA	Whether the pool should be prefilled.
pool-use-strict-min	Non-XA, XA	Whether min-pool-size should be considered strictly.

Attribute	Datasource Type	Description
prepared-statements-cache-size	Non-XA, XA	The number of prepared statements per connection in a Least Recently Used (LRU) cache.
query-timeout	Non-XA, XA	The timeout for queries, in seconds. The default is no timeout.
reauth-plugin-class-name	Non-XA, XA	The fully-qualified class name of the reauthentication plugin implementation to reauthenticate physical connections.
reauth-plugin-properties	Non-XA, XA	The properties for the reauthentication plugin.
recovery-authentication-context	XA Only	The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.
recovery-credential-reference	XA Only	Credential, from a credential store, to authenticate on datasource.
recovery-elytron-enabled	XA Only	Enables Elytron security for handling authentication of connections for recovery. The Elytron authentication-context used will be the current context if no authentication-context is specified. See authentication-context for additional information.
recovery-password	XA Only	The password to use to connect to the resource for recovery.
recovery-plugin-class-name	XA Only	The fully-qualified class name of the recovery plugin implementation.
recovery-plugin-properties	XA Only	The properties for the recovery plugin.
recovery-security-domain	XA Only	The security domain to use to connect to the resource for recovery.
recovery-username	XA Only	The user name to use to connect to the resource for recovery.
same-rm-override	XA Only	Whether the javax.transaction.xa.XAResource.isSameRM(XAResource) class returns true or false .

Attribute	Datasource Type	Description
security-domain	Non-XA, XA	The name of a JAAS security-manager which handles authentication. This name correlates to the application-policy/name attribute of the JAAS login configuration.
set-tx-query-timeout	Non-XA, XA	Whether to set the query timeout based on the time remaining until transaction timeout. Any configured query timeout will be used if no transaction exists.
share-prepared-statements	Non-XA, XA	Whether JBoss EAP should cache, instead of close or terminate, the underlying physical statement when the wrapper supplied to the application is closed by application code. The default is false .
spy	Non-XA, XA	Enable spy functionality on the JDBC layer. This logs all JDBC traffic to the datasource. Note that the logging category jboss.jdbc.spy must also be set to the log level DEBUG in the logging subsystem.
stale-connection-checker-class-name	Non-XA, XA	An instance of org.jboss.jca.adapters.jdbc.StaleConnectionChecker that provides an isStaleConnection(SQLException) method. If this method returns true , then the exception is wrapped in an org.jboss.jca.adapters.jdbc.StaleConnectionException .
stale-connection-checker-properties	Non-XA, XA	The stale connection checker properties.
statistics-enabled	Non-XA, XA	Whether runtime statistics are enabled. The default is false .
track-statements	Non-XA, XA	Whether to check for unclosed statements when a connection is returned to a pool and a statement is returned to the prepared statement cache. If false, statements are not tracked. Valid values: <ul style="list-style-type: none"> ● true: Statements and result sets are tracked, and a warning is issued if they are not closed. ● false: Neither statements or result sets are tracked. ● nowarn: Statements are tracked but no warning is issued (<i>default</i>).

Attribute	Datasource Type	Description
tracking	Non-XA, XA	Whether to track connection handles across transaction boundaries.
transaction-isolation	Non-XA, XA	<p>The java.sql.Connection transaction isolation level. Valid values:</p> <ul style="list-style-type: none"> • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_READ_COMMITTED • TRANSACTION_REPEATABLE_READ • TRANSACTION_SERIALIZABLE • TRANSACTION_NONE
url-delimiter	Non-XA, XA	The delimiter for URLs in connection-url for High Availability (HA) datasources.
url-property	XA Only	The property for the URL property in the xa-datasource-property values.
url-selector-strategy-class-name	Non-XA, XA	A class that implements org.jboss.jca.adapters.jdbc.URLSelectorStrategy .
use-ccm	Non-XA, XA	Enable the cached connection manager.
use-fast-fail	Non-XA, XA	If true, fail a connection allocation on the first attempt if the connection is invalid. If false, keep trying until the pool is exhausted.
use-java-context	Non-XA, XA	Whether to bind the datasource into global JNDI.
use-try-lock	Non-XA, XA	A timeout value for internal locks. This attempts to obtain the lock for the configured number of seconds, before timing out, rather than failing immediately if the lock is unavailable. Uses tryLock() instead of lock() .
user-name	Non-XA, XA	The user name to use when creating a new connection.

Attribute	Datasource Type	Description
valid-connection-checker-class-name	Non-XA, XA	An implementation of org.jboss.jca.adapters.jdbc.ValidConnectionChecker which provides a SQLException.isValidConnection(Connection e) method to validate a connection. An exception means the connection is destroyed. This overrides the attribute check-valid-connection-sql if it is present.
valid-connection-checker-properties	Non-XA, XA	The valid connection checker properties.
validate-on-match	Non-XA, XA	Whether connection validation is performed when a connection factory attempts to match a managed connection. This should be used when a client must have a connection validated prior to use. Validate-on-match is typically not to be used with background-validation or there will be redundant checks.
wrap-xa-resource	XA Only	Whether to wrap the XAResource in an org.jboss.tm.XAResourceWrapper instance.
xa-datasource-class	XA Only	The fully-qualified name of the javax.sql.XADataSource implementation class.
xa-datasource-properties	XA Only	String name/value pair of XA datasource properties.
xa-resource-timeout	XA Only	If non-zero, this value is passed to the XAResource.setTimeout method.

Table A.45. JDBC Driver Attributes

Attribute	Datasource Type	Description
datasource-class-info	Non-XA, XA	The available properties for the datasource-class and xa-datasource-class for the jdbc-driver . The datasource-class and xa-datasource-class attributes define the fully qualified class name that implements javax.sql.DataSource or javax.sql.XADataSource classes. The class defined can have setters for various properties. The datasource-class-info attribute lists these properties that can be set for the class.

A.21. DATASOURCE STATISTICS

Table A.46. Core Pool Statistics

Name	Description
ActiveCount	The number of active connections. Each of the connections is either in use by an application or available in the pool.
AvailableCount	The number of available connections in the pool.
AverageBlockingTime	The average time spent blocking on obtaining an exclusive lock on the pool. This value is in milliseconds.
AverageCreationTime	The average time spent creating a connection. This value is in milliseconds.
AverageGetTime	The average time spent obtaining a connection.
AveragePoolTime	The average time that a connection spent in the pool.
AverageUsageTime	The average time spent using a connection.
BlockingFailureCount	The number of failures trying to obtain a connection.
CreatedCount	The number of connections created.
DestroyedCount	The number of connections destroyed.
IdleCount	The number of connections that are currently idle.
InUseCount	The number of connections currently in use.
MaxCreationTime	The maximum time it took to create a connection. This value is in milliseconds.
MaxGetTime	The maximum time for obtaining a connection.
MaxPoolTime	The maximum time for a connection in the pool.
MaxUsageTime	The maximum time using a connection.
MaxUsedCount	The maximum number of connections used.
MaxWaitCount	The maximum number of requests waiting for a connection at the same time.
MaxWaitTime	The maximum time spent waiting for an exclusive lock on the pool.

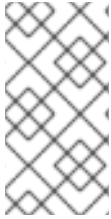
Name	Description
TimedOut	The number of timed out connections.
TotalBlockingTime	The total time spent waiting for an exclusive lock on the pool. This value is in milliseconds.
TotalCreationTime	The total time spent creating connections. This value is in milliseconds.
TotalGetTime	The total time spent obtaining connections.
TotalPoolTime	The total time spent by connections in the pool.
TotalUsageTime	The total time spent using connections.
WaitCount	The number of requests that had to wait to obtain a connection.
XACommitAverageTime	The average time for an XAResource commit invocation.
XACommitCount	The number of XAResource commit invocations.
XACommitMaxTime	The maximum time for an XAResource commit invocation.
XACommitTotalTime	The total time for all XAResource commit invocations.
XAEndAverageTime	The average time for an XAResource end invocation.
XAEndCount	The number of XAResource end invocations.
XAEndMaxTime	The maximum time for an XAResource end invocation.
XAEndTotalTime	The total time for all XAResource end invocations.
XAForgetAverageTime	The average time for an XAResource forget invocation.
XAForgetCount	The number of XAResource forget invocations.
XAForgetMaxTime	The maximum time for an XAResource forget invocation.
XAForgetTotalTime	The total time for all XAResource forget invocations.
XAPrepareAverageTime	The average time for an XAResource prepare invocation.
XAPrepareCount	The number of XAResource prepare invocations.
XAPrepareMaxTime	The maximum time for an XAResource prepare invocation.

Name	Description
XAPrepareTotalTime	The total time for all XAResource prepare invocations.
XARecoverAverageTime	The average time for an XAResource recover invocation.
XARecoverCount	The number of XAResource recover invocations.
XARecoverMaxTime	The maximum time for an XAResource recover invocation.
XARecoverTotalTime	The total time for all XAResource recover invocations.
XARollbackAverageTime	The average time for an XAResource rollback invocation.
XARollbackCount	The number of XAResource rollback invocations.
XARollbackMaxTime	The maximum time for an XAResource rollback invocation.
XARollbackTotalTime	The total time for all XAResource rollback invocations.
XAStartAverageTime	The average time for an XAResource start invocation.
XAStartCount	The number of XAResource start invocations.
XAStartMaxTime	The maximum time for an XAResource start invocation.
XAStartTotalTime	The total time for all XAResource start invocations.

Table A.47. JDBC Statistics

Name	Description
PreparedStatementCacheAccessCount	The number of times that the statement cache was accessed.
PreparedStatementCacheAddCount	The number of statements added to the statement cache.
PreparedStatementCacheCurrentSize	The number of prepared and callable statements currently cached in the statement cache.
PreparedStatementCacheDeleteCount	The number of statements discarded from the cache.
PreparedStatementCacheHitCount	The number of times that statements from the cache were used.
PreparedStatementCacheMissCount	The number of times that a statement request could not be satisfied with a statement from the cache.

A.22. AGROAL DATASOURCE ATTRIBUTES



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-agroal_1_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.48. Agroal Datasource Attributes

Attribute	Description
connectable	Whether to enable CMR (Commit Markable Resource) functionality on this datasource. This applies to non-XA datasources only.
jndi-name	Specifies the JNDI name for the datasource.
jta	Whether to enable JTA integration. This applies to non-XA datasources only.
statistics-enabled	Whether to enable statistics for this datasource. Defaults to false .

Table A.49. Agroal Datasource Connection Factory Attributes

Attribute	Description
authentication-context	Reference to an authentication context in the elytron subsystem.
connection-properties	Properties to be passed to the JDBC driver when creating a connection.
credential-reference	Credential, from a credential store, to authenticate with.
driver	A unique reference to the JDBC driver.
new-connection-sql	A SQL statement to be executed on a connection after creation.
password	The password to use for basic authentication with the database.
transaction-isolation	Set the java.sql.Connection transaction isolation level to used.
url	The JDBC driver connection URL.

Attribute	Description
username	The username to use for basic authentication with the database.

Table A.50. Agroal Datasource Connection Pool Attributes

Attribute	Description
background-validation	The time, in milliseconds, between background validation runs.
blocking-timeout	The maximum time, in milliseconds, to block while waiting for a connection before throwing an exception.
idle-removal	The time, in minutes, that a connection must be idle before it can be removed.
initial-size	The initial number of connections the pool should hold.
leak-detection	The time, in milliseconds, that a connection must be held before a leak warning.
max-size	The maximum number of connections in the pool.
min-size	The minimum number of connections the pool should hold.

A.23. TRANSACTION MANAGER CONFIGURATION OPTIONS



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-txn_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.51. Transactions Subsystem Attributes

Attribute	Description
default-timeout	The default transaction timeout. This defaults to 300 seconds. You can override this programmatically, on a per-transaction basis.
enable-statistics	<i>Deprecated in favor of statistics-enabled.</i>

Attribute	Description
enable-tsm-status	Whether to enable the transaction status manager (TSM) service, which is used for out-of-process recovery. This option is not supported, as running an out-of-process recovery manager to contact the ActionStatusService from a different process, instead of in memory, is not supported.
hornetq-store-enable-async-io	<i>Deprecated in favor of journal-store-enable-async-io.</i>
jdbc-action-store-drop-table	Whether JDBC action store should drop tables. The default is false .
jdbc-action-store-table-prefix	Optional prefix for table used to write transaction logs in configured JDBC action store.
jdbc-communication-store-drop-table	Whether JDBC communication store should drop tables. The default is false .
jdbc-communication-store-table-prefix	Optional prefix for table used to write transaction logs in configured JDBC communication store.
jdbc-state-store-drop-table	Whether JDBC state store should drop tables. The default is false .
jdbc-state-store-table-prefix	Optional prefix for table used to write transaction logs in configured JDBC state store.
jdbc-store-datasource	JNDI name of non-XA datasource used. Datasource should be defined in the datasources subsystem.
journal-store-enable-async-io	Whether AsyncIO should be enabled for the journal store or not. Defaults to false . The server should be restarted for this setting to take effect.
jts	Whether to use Java Transaction Service (JTS) transactions. Defaults to false , which uses JTA transactions only.
maximum-timeout	If a transaction is set to have a transaction timeout of 0 , which implies an unlimited timeout, the transaction manager uses the value set by this attribute instead. The default is 31536000 seconds (365 days).

Attribute	Description
node-identifier	<p>The node identifier for the transaction manager. If this option is not set, you will see a warning upon server startup. This option is required in the following situations:</p> <ul style="list-style-type: none"> • For JTS to JTS communications • When two transaction managers access shared resource managers • When two transaction managers access shared object stores <p>The node-identifier must be unique for each transaction manager as it is required to enforce data integrity during recovery. The node-identifier must also be unique for JTA because multiple nodes may interact with the same resource manager or share a transaction object store.</p>
object-store-path	<p>A relative or absolute file system path where the transaction manager object store stores data. By default relative to the object-store-relative-to parameter's value. If object-store-relative-to is set to an empty string, this value is treated as an absolute path.</p>
object-store-relative-to	<p>References a global path configuration in the domain model. The default value is the data directory for JBoss EAP, which is the value of the property jboss.server.data.dir, and defaults to EAP_HOME/domain/data/ for a managed domain, or EAP_HOME/standalone/data/ for a standalone server instance. The value of the object store object-store-path transaction manager attribute is relative to this path. Set this attribute to an empty string to have object-store-path be treated as an absolute path.</p>
process-id-socket-binding	<p>The name of the socket binding configuration to use if the transaction manager should use a socket-based process ID. Will be undefined if process-id-uuid is true; otherwise must be set.</p>
process-id-socket-max-ports	<p>The transaction manager creates a unique identifier for each transaction log. Two different mechanisms are provided for generating unique identifiers: a socket-based mechanism and a mechanism based on the process identifier of the process.</p> <p>In the case of the socket-based identifier, a socket is opened and its port number is used for the identifier. If the port is already in use, the next port is probed, until a free one is found. The process-id-socket-max-ports represents the maximum number of sockets the transaction manager will try before failing. The default value is 10.</p>
process-id-uuid	<p>Set to true to use the process identifier to create a unique identifier for each transaction. Otherwise, the socket-based mechanism is used. Defaults to true. See process-id-socket-max-ports for more information. To enable process-id-socket-binding, set process-id-uuid to false.</p>

Attribute	Description
recovery-listener	Whether or not the transaction recovery process should listen on a network socket. Defaults to false .
socket-binding	Specifies the name of the socket binding used by the transaction periodic recovery listener when recovery-listener is set to true .
statistics-enabled	Whether statistics should be enabled. The default is false .
status-socket-binding	Specifies the socket binding to use for the transaction status manager. This configuration option is not supported.
use-hornetq-store	<i>Deprecated in favor of use-journal-store.</i>
use-jdbc-store	Use the JDBC store for writing transaction logs. Set to true to enable and to false to use the default log store type.
use-journal-store	Use Apache ActiveMQ Artemis journaled storage mechanisms instead of file-based storage for the transaction logs. This is disabled by default, but can improve I/O performance. It is not recommended for JTS transactions on separate transaction managers. When changing this option, the server has to be restarted using the shutdown command for the change to take effect.

Table A.52. Log Store Attributes

Attribute	Description
expose-all-logs	Whether to expose all logs. The default is false , meaning that only a subset of transaction logs is exposed.
type	Specifies the implementation type of the logging store. The default is default .

Table A.53. Commit Markable Resource Attributes

Attribute	Description
batch-size	The batch size for this CMR resource. The default is 100 .
immediate-cleanup	Whether to perform immediate cleanup for this CMR resource. The default is true .
jndi-name	The JNDI name of this CMR resource.

Attribute	Description
name	The table name for storing XIDs. The default is xids .

A.24. IIOP SUBSYSTEM ATTRIBUTES



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-iiop-openjdk_3_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.54. IIOP Subsystem Attributes

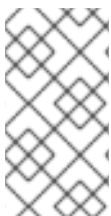
Attribute	Description
add-component-via-interceptor	Indicates whether SSL components should be added by an IOR interceptor. <i>Deprecated.</i>
auth-method	The authentication method. Valid values are none and username_password .
authentication-context	The name of the authentication context used when the security initializer is set to elytron .
caller-propagation	Indicates whether the caller identity should be propagated in the SAS context. Valid values are none and supported .
client-requires	Value that indicates the client SSL required parameters. Valid values are None , ServerAuth , ClientAuth , and MutualAuth . <i>Deprecated: Use client-requires-ssl instead.</i>
client-requires-ssl	Indicates whether IIOP connections from the server require SSL.
client-ssl-context	The name of the SSL context used to create client-side SSL sockets.
client-supports	Value that indicates the client SSL supported parameters. Valid values are None , ServerAuth , ClientAuth , and MutualAuth . <i>Deprecated: Use client-requires-ssl instead.</i>
confidentiality	Indicates whether the transport must require confidentiality protection or not. Valid values are none , supported , and required . <i>Deprecated: Use server-requires-ssl instead.</i>

Attribute	Description
detect-misordering	Indicates whether the transport must require misordering detection or not. Valid values are none , supported , and required . <i>Deprecated: Use server-requires-ssl instead.</i>
detect-replay	Indicates whether the transport must require replay detection or not. Valid values are none , supported , and required . <i>Deprecated: Use server-requires-ssl instead.</i>
export-corballoc	Indicates whether the root context should be exported as corballoc::address:port/NameService .
giop-version	The GIOP version to be used.
high-water-mark	TCP connection cache parameter. Each time the number of connections exceeds this value, the ORB tries to reclaim connections. The number of reclaimed connections is specified by the number-to-reclaim property. If this property is not set, then the OpenJDK ORB default is used.
integrity	Indicates whether the transport must require integrity protection or not. Valid values are none , supported , and required . <i>Deprecated: Use server-requires-ssl instead.</i>
number-to-reclaim	TCP connection cache parameter. Each time the number of connections exceeds the high-water-mark property, then the ORB tries to reclaim connections. The number of reclaimed connections is specified by this property. If it is not set, then the OpenJDK ORB default is used.
persistent-server-id	Persistent ID of the server. Persistent object references are valid across many activations of the server and they identify it using this property. As a result of that, many activations of the same server should have this property set to the same value, and different server instances running on the same host should have different server IDs.
properties	A list of generic key/value properties.
realm	The authentication service realm name.
required	Indicates whether authentication is required.
root-context	The naming service root context.
security	Indicates whether the security interceptors are to be installed. Valid values are client , identity , elytron , and none .

Attribute	Description
security-domain	The name of the security domain that holds the keystores and truststores that will be used to establish SSL connections.
server-requires	Value that indicates the server SSL required parameters. Valid values are None , ServerAuth , ClientAuth , and MutualAuth . <i>Deprecated: Use server-requires-ssl instead.</i>
server-requires-ssl	Indicates whether IIOP connections to the server require SSL.
server-ssl-context	The name of the SSL context used to create server-side SSL sockets.
server-supports	Value that indicates the server SSL supported parameters. Valid values are None , ServerAuth , ClientAuth , and MutualAuth . <i>Deprecated: Use server-requires-ssl instead.</i>
socket-binding	The name of the socket binding configuration that specifies the ORB port.
ssl-socket-binding	The name of the socket binding configuration that specifies the ORB SSL port.
support-ssl	Indicates whether SSL is supported.
transactions	Indicates whether the transactions interceptors are to be installed or not. Valid values are full , spec , and none . A value of full enables JTS while a value of spec enables a non-JTS spec-compliant mode that rejects incoming transaction contexts.
trust-in-client	Indicates if the transport must require trust in client to be established. Valid values are none , supported , and required . <i>Deprecated: Use server-requires-ssl instead.</i>
trust-in-target	Indicates if the transport must require trust in target to be established. Valid values are none and supported . <i>Deprecated: Use server-requires-ssl instead.</i>

A.25. RESOURCE ADAPTER ATTRIBUTES

The following tables describe the resource adapter attributes.



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-resource-adapters_5_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.55. Main Attributes

Attribute	Description
archive	The resource adapter archive.
beanvalidationgroups	The bean validation groups that should be used.
bootstrap-context	The unique name of the bootstrap context that should be used.
config-properties	Custom defined config properties.
module	The module from which the resource adapter will be loaded.
statistics-enabled	Whether runtime statistics are enabled or not.
transaction-support	The transaction support level of the resource adapter. Valid values are NoTransaction , LocalTransaction , or XATransaction .
wm-elytron-security-domain	Defines the name of the Elytron security domain that should be used.
wm-security	Toggle on/off wm.security for this resource adapter. In case of false, all wm-security -* parameters are ignored, even the defaults.
wm-security-default-groups	A default groups list that should be added to the used Subject instance.
wm-security-default-principal	A default principal name that should be added to the used Subject instance.
wm-security-domain	The name of the security domain that should be used.
wm-security-mapping-groups	List of groups mappings.
wm-security-mapping-required	Defines if a mapping is required for security credentials.
wm-security-mapping-users	List of user mappings.

**NOTE**

If your resource adapter is using **bootstrap-context** along with a work manager that has **elytron-enabled** set to **true**, you must use the **wm-elytron-security-domain** attribute instead of the **wm-security-domain** attribute for security domain specification.

Table A.56. admin-objects Attributes

Attribute	Description
class-name	The fully qualified class name of an administration object.
enabled	Specifies if the administration object should be enabled.
jndi-name	The JNDI name for the administration object.
use-java-context	Setting this to false will bind the object into global JNDI.

Table A.57. connection-definitions Attributes

Attribute	Description
allocation-retry	Indicates the number of times that allocating a connection should be tried before throwing an exception.
allocation-retry-wait-millis	The amount of time, in milliseconds, to wait between retrying to allocate a connection.
authentication-context	The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.
authentication-context-and-application	Indicates that either application-supplied parameters, such as from getConnection(user, pw) , or Subject , are used to distinguish connections in the pool. These parameters are provided by Elytron after authentication when using a configured authentication-context .
background-validation	Specifies that connections should be validated on a background thread versus being validated prior to use. Changing this value requires a server restart.
background-validation-millis	The amount of time, in milliseconds, that background validation will run. Changing this value requires a server restart.
blocking-timeout-wait-millis	The maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time.
capacity-decrementer-class	Class defining the policy for decrementing connections in the pool.
capacity-decrementer-properties	Properties to inject in class defining the policy for decrementing connections in the pool.
capacity-incrementer-class	Class defining the policy for incrementing connections in the pool.

Attribute	Description
capacity-incrementer-properties	Properties to inject in class defining the policy for incrementing connections in the pool.
class-name	The fully qualified class name of a managed connection factory or admin object.
connectable	Enable the use of CMR. This feature means that a local resource can reliably participate in an XA transaction.
elytron-enabled	Enables Elytron security for handling authentication of connections. The Elytron authentication-context to be used will be the current context if no context is specified. See authentication-context for additional information.
enabled	Specifies if the resource adapter should be enabled.
enlistment	Specifies if lazy enlistment should be used if supported by the resource adapter.
enlistment-trace	Specifies if JBoss EAP/IronJacamar should record enlistment traces. This is false by default.

Attribute	Description
flush-strategy	<p>Specifies how the pool should be flushed in case of an error. Valid values are:</p> <p>FailingConnectionOnly Only the failing connection is removed. This is the default setting.</p> <p>InvalidIdleConnections The failing connection and idle connections that share the same credentials and are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed.</p> <p>IdleConnections The failing connection and idle connections that share the same credentials are removed.</p> <p>Gracefully The failing connection and idle connections that share the same credentials are removed. Active connections that share the same credentials are destroyed upon return to the pool.</p> <p>EntirePool The failing connection and idle and active connections that share the same credentials are removed. This setting is not recommended for production systems.</p> <p>AllInvalidIdleConnections The failing connection and idle connections that are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed.</p> <p>AllIdleConnections The failing connection and all idle connections are removed.</p> <p>AllGracefully The failing connection and all idle connections are removed. Active connections are destroyed upon return to the pool.</p> <p>AllConnections The failing connection and all idle and active connections are removed. This setting is not recommended for production systems.</p>
idle-timeout-minutes	<p>The maximum time, in minutes, a connection may be idle before being closed. The actual maximum time depends also on the IdleRemover scan time, which is half of the smallest idle-timeout-minutes value of any pool. Changing this value requires a server restart.</p>
initial-pool-size	<p>The initial number of connections a pool should hold.</p>
interleaving	<p>Specifies whether to enable interleaving for XA connections.</p>
jndi-name	<p>The JNDI name for the connection factory.</p>
max-pool-size	<p>The maximum number of connections for a pool. No more connections will be created in each sub-pool.</p>

Attribute	Description
mcp	The ManagedConnectionPool implementation. For example: org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool .
min-pool-size	The minimum number of connections for a pool.
no-recovery	Specifies if the connection pool should be excluded from recovery.
no-tx-separate-pool	Oracle does not like XA connections getting used both inside and outside a JTA transaction. To workaround the problem you can create separate sub-pools for the different contexts.
pad-xid	Specifies whether the Xid should be padded.
pool-fair	Specifies if pool use should be fair.
pool-prefill	Specifies if the pool should be prefilled. Changing this value requires a server restart.
pool-use-strict-min	Specifies if the min-pool-size should be considered strict.
recovery-authentication-context	The Elytron authentication context used for recovery. If no authentication-context is specified, then the current one will be used.
recovery-credential-reference	Credential, from a credential store, to authenticate on recovery of the connection.
recovery-elytron-enabled	Indicates that an Elytron authentication context will be used for recovery. The default is false .
recovery-password	The password used for recovery.
recovery-plugin-class-name	The fully qualified class name of the recovery plugin implementation.
recovery-plugin-properties	The properties for the recovery plugin.
recovery-security-domain	The security domain used for recovery.
recovery-username	The user name used for recovery.
same-rm-override	Unconditionally set whether javax.transaction.xa.XAResource.isSameRM(XAResource) returns true or false.

Attribute	Description
security-application	Indicates that application-supplied parameters, such as from getConnection(user, pw) , are used to distinguish connections in the pool.
security-domain	The security domain which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.
security-domain-and-application	Indicates that either application-supplied parameters, such as from getConnection(user, pw) , or Subject , from the security domain, are used to distinguish connections in the pool.
sharable	Enable the use of sharable connections, which allows lazy association to be enabled if supported.
tracking	Specifies if IronJacamar should track connection handles across transaction boundaries.
use-ccm	Enable the use of a cached connection manager.
use-fast-fail	When set to true , fail a connection allocation on the first try if it is invalid. When set to false , keep trying until the pool is exhausted of all potential connections.
use-java-context	Setting this to false will bind the object into global JNDI.
validate-on-match	Specifies if connection validation should be done when a connection factory attempts to match a managed connection. This is typically exclusive to the use of background validation.
wrap-xa-resource	Specifies whether XAResource instances should be wrapped in an org.jboss.tm.XAResourceWrapper instance.
xa-resource-timeout	The value is passed to XAResource.setTimeout() , in seconds. The default is 0 .

A.26. RESOURCE ADAPTER STATISTICS

Table A.58. Resource Adapter Statistics

Name	Description
ActiveCount	The number of active connections. Each of the connections is either in use by an application or available in the pool
AvailableCount	The number of available connections in the pool.

Name	Description
AverageBlockingTime	The average time spent blocking on obtaining an exclusive lock on the pool. The value is in milliseconds.
AverageCreationTime	The average time spent creating a connection. The value is in milliseconds.
CreatedCount	The number of connections created.
DestroyedCount	The number of connections destroyed.
InUseCount	The number of connections currently in use.
MaxCreationTime	The maximum time it took to create a connection. The value is in milliseconds.
MaxUsedCount	The maximum number of connections used.
MaxWaitCount	The maximum number of requests waiting for a connection at the same time.
MaxWaitTime	The maximum time spent waiting for an exclusive lock on the pool.
TimedOut	The number of timed out connections.
TotalBlockingTime	The total time spent waiting for an exclusive lock on the pool. The value is in milliseconds.
TotalCreationTime	The total time spent creating connections. The value is in milliseconds.
WaitCount	The number of requests that had to wait for a connection.

A.27. UNDERTOW SUBSYSTEM ATTRIBUTES

See the tables below for the attributes of the various elements of the **undertow** subsystem.



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-undertow_4_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

- [Main Attributes](#)
- [Application Security Domain Attributes](#)
- [Buffer Cache Attributes](#)

- [Byte Buffer Pool Attributes](#)
- [Servlet Container Attributes](#)
- [Filter Attributes](#)
- [Handler Attributes](#)
- [Server Attributes](#)

Table A.59. Main undertow Attributes

Attribute	Default	Description
default-security-domain	other	The default security domain used by web deployments.
default-server	default-server	The default server to use for deployments.
default-servlet-container	default	The default servlet container to use for deployments.
default-virtual-host	default-host	The default virtual host to use for deployments.
instance-id	<code>\${jboss.node.name}</code>	The cluster instance ID.
statistics-enabled	false	Whether statistics are enabled.

Application Security Domain Attributes

The application security domain attributes has the following structure:

- **application-security-domain**
 - **setting**
 - **single-sign-on**

application-security-domain Attributes

Table A.60. application-security-domain Attributes

Attribute	Default	Description
enable-jacc	false	Enable authorization using JACC.
enable-jaspi	true	Enable JASPI authentication for the associated deployments.
http-authentication-factory		The HTTP authentication factory to be used by deployments that reference the mapped security domain.

Attribute	Default	Description
integrated-jaspi	true	Whether integrated JASPI should be used. When set to true during JASPI authentication, the identity is loaded from the SecurityDomain referenced by the deployment. When set to false , an ad hoc identity is created instead.
override-deployment-config	false	Whether the authentication configuration in the deployment should be overridden by the factory.
referencing-deployments		The deployments currently referencing this mapping.
security-domain		The SecurityDomain to be used by the deployments.

single-sign-on Attributes

Table A.61. single-sign-on Attributes

Attribute	Default	Description
client-ssl-context		Reference to the SSL context used to secure back-channel logout connection.
cookie-name	JSESSIONIDS SO	Name of the cookie.
credential-reference		The credential reference to decrypt the private key entry.
domain		The cookie domain that will be used.
http-only	false	Set cookie httpOnly attribute.
key-alias		Alias of the private key entry used for signing and verifying back-channel logout connection.
key-store		Reference to keystore containing a private key entry.
path	/	Cookie path.
secure	false	Set cookie secure attribute.

Buffer Cache Attributes

Table A.62. buffer-cache Attributes

Attribute	Default	Description
buffer-size	1024	The size of the buffers. Smaller buffers allow space to be utilized more effectively.
buffers-per-region	1024	The numbers of buffers per region.
max-regions	10	The maximum number of regions. This controls the maximum amount of memory that can be used for caching.

Byte Buffer Pool Attributes

Table A.63. byte-buffer-pool Attributes

Attribute	Default	Description
buffer-size		<p>The size, in bytes, of each buffer slice. If not specified, the size is set based on the available RAM of your system:</p> <ul style="list-style-type: none"> • 512 bytes for less than 64 MB RAM • 1024 bytes (1 KB) for 64 MB - 128 MB RAM • 16384 bytes (16 KB) for more than 128 MB RAM <p>For performance tuning advice on this attribute, see Configuring Buffer Pools in the JBoss EAP <i>Performance Tuning Guide</i>.</p>
direct		<p>Boolean value that denotes if this buffer is a direct or heap pool. If not specified, the value is set based on the available RAM of your system:</p> <ul style="list-style-type: none"> • If available RAM is < 64MB, the value is set to false • If available RAM is >= 64MB, the value is set to true <p>Note that direct pools also have a corresponding heap pool.</p>
leak-detection-percent	0	The percentage of buffers that should be allocated with a leak detector.
max-pool-size		The maximum number of buffers to keep in the pool. Buffers will still be allocated above this limit, but will not be retained if the pool is full.

Attribute	Default	Description
thread-local-cache-size	12	The size of the per-thread cache. This is a maximum size, the cache will use smart sizing to only keep buffers on the thread if the thread is actually allocating buffers.

Servlet Container Attributes

The servlet container component has the following structure:

- **servlet-container**
 - **mime-mapping**
 - **setting**
 - **crawler-session-management**
 - **jsp**
 - **persistent-sessions**
 - **session-cookie**
 - **websockets**
 - **welcome-file**

servlet-container Attributes

Table A.64. servlet-container Attributes

Attribute	Default	Description
allow-non-standard-wrappers	false	Whether request and response wrappers that do not extend the standard wrapper classes can be used.
default-buffer-cache	default	The buffer cache to use for caching static resources.
default-cookie-version	0	The default cookie version to use for cookies created by the application.
default-encoding		Default encoding to use for all deployed applications.
default-session-timeout	30	The default session timeout in minutes for all applications deployed in the container.
directory-listing		If directory listing should be enabled for default servlets.

Attribute	Default	Description
disable-caching-for-secured-pages	true	Whether to set headers to disable caching for secured paged. Disabling this can cause security problems, as sensitive pages may be cached by an intermediary.
disable-file-watch-service	false	If set to true , then the file watch service will not be used to monitor exploded deployments for changes. This attribute overrides the io.undertow.disable-file-system-watcher system property.
disable-session-id-reuse	false	If set to true , then an unknown session ID will never be reused and a new session ID will be generated. If set to false , then the session ID will be reused only if it is present in the session manager of another deployment to allow the same session ID to be shared between applications on the same server.
eager-filter-initialization	false	Whether to call filter init() on deployment start rather than when first requested.
ignore-flush	false	Ignore flushes on the servlet output stream. In most cases these just hurt performance for no good reason.
max-sessions		The maximum number of sessions that can be active at one time.
proactive-authentication	true	Whether proactive authentication should be used. If this is true , a user will always be authenticated if credentials are present.
session-id-length	30	Longer session ID's are more secure. This value specifies the length of the generated session ID in bytes. The system encodes the generated session ID as a Base64 string and provides the result to the client as a session ID cookie. As a result of this processing, the server sends to the client a cookie value that is approximately 33% larger than the session ID that it originally generated. For example, a session ID length of 30 results in a cookie value length of 40.
stack-trace-on-error	local-only	If an error page with the stack trace should be generated on error. Values are all , none and local-only .
use-listener-encoding	false	Use encoding defined on listener.

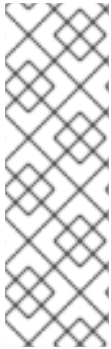
mime-mapping Attributes

Table A.65. mime-mapping Attributes

Attribute	Default	Description
value		The mime type for this mapping.

crawler-session-management Attributes

Configures special session handling for crawler bots.



NOTE

When using the management CLI to manage the **crawler-session-management** element, it is available under **settings** in the **servlet-container** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=crawler-session-  
management:add  
/subsystem=undertow/servlet-container=default/setting=crawler-session-  
management:read-resource
```

Table A.66. crawler-session-management Attributes

Attribute	Default	Description
session-timeout		The session timeout in seconds for sessions that are owned by crawlers.
user-agents		Regular expression that is used to match the user agent of a crawler.

jsp Attributes



NOTE

When using the management CLI to manage the **jsp** element, it is available under **settings** in the **servlet-container** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=jsp:read-resource
```

Table A.67. jsp Attributes

Attribute	Default	Description
check-interval	0	Check interval for JSP updates using a background thread. This has no effect for most deployments where JSP change notifications are handled using the file system notification API. This only takes effect if the file watch service is disabled.

Attribute	Default	Description
development	false	Enable development mode which enables reloading JSP on-the-fly.
disabled	false	Enable the JSP container.
display-source-fragment	true	When a runtime error occurs, attempts to display corresponding JSP source fragment.
dump-smap	false	Write SMAP data to a file.
error-on-use-bean-invalid-class-attribute	false	Enable errors when using a bad class in useBean.
generate-strings-as-char-arrays	false	Generate String constants as char arrays.
java-encoding	UTF8	Specify the encoding used for Java sources.
keep-generated	true	Keep the generated servlets.
mapped-file	true	Map to the JSP source.
modification-test-interval	4	Minimum amount of time between two tests for updates, in seconds.
optimize-scriptlets	false	If JSP scriptlets should be optimized to remove string concatenation.
recompile-on-fail	false	Retry failed JSP compilations on each request.
scratch-dir		Specify a different work directory.
smap	true	Enable SMAP.
source-vm	1.8	Source VM level for compilation.
tag-pooling	true	Enable tag pooling.
target-vm	1.8	Target VM level for compilation.
trim-spaces	false	Trim some spaces from the generated servlet.
x-powered-by	true	Enable advertising the JSP engine in x-powered-by.

persistent-sessions Attributes

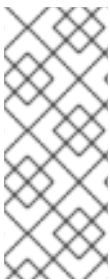
**NOTE**

When using the management CLI to manage the **persistent-sessions** element, it is available under **settings** in the **servlet-container** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=persistent-sessions:add
/subsystem=undertow/servlet-container=default/setting=persistent-sessions:read-resource
```

Table A.68. persistent-sessions Attributes

Attribute	Default	Description
path		The path to the persistent session data directory. If this is null, sessions will be stored in memory.
relative-to		The directory the path is relative to.

session-cookie Attributes**NOTE**

When using the management CLI to manage the **session-cookie** element, it is available under **settings** in the **servlet-container** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:add
/subsystem=undertow/servlet-container=default/setting=session-cookie:read-resource
```

Table A.69. session-cookie Attributes

Attribute	Default	Description
comment		Cookie comment.
domain		Cookie domain.
http-only		Whether the cookie is http-only.
max-age		Maximum age of the cookie.
name		Name of the cookie.
secure		Whether the cookie is secure.

websockets Attributes

**NOTE**

When using the management CLI to manage the **websockets** element, it is available under **settings** in the **servlet-container** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=websockets:read-resource
```

Table A.70. websockets Attributes

Attribute	Default	Description
buffer-pool	default	The buffer pool to use for websocket deployments.
deflater-level	0	Configures the level of compression of the DEFLATE algorithm.
dispatch-to-worker	true	Whether callbacks should be dispatched to a worker thread. If this is false , then they will be run in the IO thread, which is faster however care must be taken not to perform blocking operations.
per-message-deflate	false	Enables websocket's per-message compression extension.
worker	default	The worker to use for websocket deployments.

welcome-file Attributes

Defines a welcome file and has no options.

Filter Attributes

These components can be found at **/subsystem=undertow/configuration=filter**.

custom-filter Filters**Table A.71. custom-filter Attributes**

Attribute	Default	Description
class-name		Class name of HttpHandler.
module		Module name where class can be loaded from.
parameters		Filter parameters.

error-page Filters

The error pages

Table A.72. error-page Attributes

Attribute	Default	Description
code		Error page code.
path		Error page path.

expression-filter Filters

A filter parsed from the Undertow expression language.

Table A.73. expression-filter Attributes

Attribute	Default	Description
expression		The expression that defines the filter.
module		Module to use to load the filter definitions.

gzip Filters

Defines the gzip filter and has no attributes.

mod-cluster Filters


The mod-cluster filter component has the following structure:

- **mod-cluster**
 - **balancer**
 - **load-balancing-group**
 - **node**
 - **context**

Table A.74. mod-cluster Attributes

Attribute	Default	Description
advertise-frequency	10000	The frequency in milliseconds that mod_cluster advertises itself on the network.
advertise-path	/	The path that mod_cluster is registered under.
advertise-protocol	http	The protocol that is in use.
advertise-socket-binding		The multicast group that is used to advertise.
broken-node-timeout	60000	The amount of time that must elapse before a broken node is removed from the table.

Attribute	Default	Description
cached-connections-per-thread	5	The number of connections that will be kept alive indefinitely.
connection-idle-timeout	60	The amount of time a connection can be idle before it will be closed. Connections will not time out once the pool size is down to the configured minimum, which is configured by cached-connections-per-thread .
connections-per-thread	10	The number of connections that will be maintained to back-end servers, per IO thread.
enable-http2	false	Whether the load balancer should attempt to upgrade back-end connections to HTTP/2. If HTTP/2 is not supported, HTTP or HTTPS will be used as normal.
failover-strategy	LOAD_BALANCED	The attribute that determines how a failover node is chosen, in the event that the node to which a session has affinity is not available.
health-check-interval	10000	The frequency of health check pings to back-end nodes.
http2-enable-push	true	Whether push should be enabled for HTTP/2 connections.
http2-header-table-size	4096	The size of the header table used for HPACK compression, in bytes. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.
http2-initial-window-size	65535	The flow control window size, in bytes, that controls how quickly the client can send data to the server.
http2-max-concurrent-streams		The maximum number of HTTP/2 streams that can be active at any time on a single connection.
http2-max-frame-size	16384	The maximum HTTP/2 frame size, in bytes.
http2-max-header-list-size		The maximum size, in bytes, of request headers the server is prepared to accept.

Attribute	Default	Description
management-access-predicate		A predicate that is applied to incoming requests to determine if they can perform mod_cluster management commands. Provides additional security on top of what is provided by limiting management to requests that originate from the management-socket-binding .
management-socket-binding		The socket binding of the mod_cluster management port. When using mod_cluster two HTTP listeners should be defined, a public one to handle requests, and one bound to the internal network to handle mod_cluster commands. This socket binding should correspond to the internal listener, and should not be publicly accessible.
max-ajp-packet-size	8192	The maximum size, in bytes, for AJP packets. Increasing this will allow AJP to work for requests and responses that have a large amount of headers. This must be the same between load balancers and backend servers.
max-request-time	-1	The maximum amount of time that a request to a back-end node can take before it is killed.
max-retries	1	<p>The number of times that an attempt to retry a request will be made, if the request fails.</p> <div>  <p>NOTE</p> <p>If a request is not considered idempotent, it will only be retried if the proxy can be sure that it was not sent to the backend server.</p> </div>
request-queue-size	10	The number of requests that can be queued if the connection pool is full before requests are rejected with a 503.
security-key		The security key that is used for the mod_cluster group. All members must use the same security key.
security-realm		The security realm that provides the SSL configuration. <i>Deprecated: Use the ssl-context attribute to reference a configured SSLContext directly.</i>
ssl-context		The reference to the SSLContext that is used by the filter.

Attribute	Default	Description
use-alias	false	Whether an alias check is performed.
worker	default	The XNIO worker that is used to send the advertise notifications.

Table A.75. balancer Attributes

Attribute	Default	Description
max-attempts		The number of attempts to send the request to a back-end server.
sticky-session		If sticky sessions are enabled.
sticky-session-cookie		The session cookie name.
sticky-session-force		If this is true , then an error will be returned if the request cannot be routed to the sticky node, otherwise it will be routed to another node.
sticky-session-path		The path of the sticky session cookie.
sticky-session-remove		Remove the session cookie if the request cannot be routed to the correct host.
wait-worker		The number of seconds to wait for an available worker.

load-balancing-group Attributes

Defines a load balancing group and has no options.

Table A.76. node Attributes

Attribute	Default	Description
aliases		The nodes aliases.
cache-connections		The number of connections to keep alive indefinitely.
elected		The elected count.
flush-packets		If received data should be immediately flushed.
load		The current load of this node.

Attribute	Default	Description
load-balancing-group		The load balancing group this node belongs to.
max-connections		The maximum number of connections per IO thread.
open-connections		The current number of open connections.
ping		The nodes ping.
queue-new-requests		If a request is received and there is no worker immediately available should it be queued.
read		The number of bytes read from the node.
request-queue-size		The size of the request queue.
status		The current status of this node.
timeout		The request timeout.
ttl		The time connections will stay alive with no requests before being closed, if the number of connections is larger than cache-connections .
uri		The URI that the load balancer uses to connect to the node.
written		The number of bytes transferred to the node.

Table A.77. context Attributes

Attribute	Default	Description
requests		The number of requests against this context.
status		The status of this context.

request-limit Filters

Table A.78. request-limit Attributes

Attribute	Default	Description
max-concurrent-requests		Maximum number of concurrent requests.

Attribute	Default	Description
queue-size		Number of requests to queue before they start being rejected.

response-header Filters

Response header filter allows you to add custom headers.

Table A.79. response-header Attributes

Attribute	Default	Description
header-name		The header name.
header-value		The header value.

rewrite Filters

Table A.80. rewrite Attributes

Attribute	Default	Description
redirect	false	Whether a redirect will be done instead of a rewrite.
target		The expression that defines the target. If you are redirecting to a constant target put single quotes around the value.

Handler Attributes

These components can be found at `/subsystem=undertow/configuration=handler`.

file Attributes

Table A.81. file Attributes

Attribute	Default	Description
cache-buffer-size	1024	Size of the buffers.
cache-buffers	1024	Number of buffers.
case-sensitive	true	Whether to use case-sensitive file handling. Note that setting this to false for case insensitivity will only work if the underlying file system is case insensitive.
directory-listing	false	Whether to enable directory listing.
follow-symlink	false	Whether to enable following symbolic links.

Attribute	Default	Description
path		Path on the file system from where file handler will serve resources.
safe-symlink-paths		Paths that are safe to be targets of symbolic links.

Using WebDAV for Static Resources


Previous versions of JBoss EAP allowed for using WebDAV with the **web** subsystem, by way of the **WebdavServlet**, to host static resources and enable additional HTTP methods for accessing and manipulating those files. In JBoss EAP 7, the **undertow** subsystem does provide a mechanism for serving static files using a file handler, but the **undertow** subsystem does not support WebDAV. If you want to use WebDAV with JBoss EAP 7, you can write a custom WebDAV servlet.

reverse-proxy attributes

The reverse-proxy handler component has the following structure:

- **reverse-proxy**
 - **host**

Table A.82. reverse-proxy Attributes

Attribute	Default	Description
cached-connections-per-thread	5	The number of connections that will be kept alive indefinitely.
connection-idle-timeout	60	The amount of time a connection can be idle before it will be closed. Connections will not time out once the pool size is down to the configured minimum (as configured by cached-connections-per-thread).
connections-per-thread	40	The number of connections that will be maintained to back-end servers, per IO thread.
max-request-time	-1	The maximum time that a proxy request can be active for, before being killed. Defaults to unlimited.
max-retries	1	<div> <div>The number of times that an attempt to retry a request will be made, if the request fails.</div> <div>  <div> NOTE <p>If a request is not considered idempotent, it will only be retried if the proxy can be sure that it was not sent to the backend server.</p> </div> </div> </div>

Attribute	Default	Description
problem-server-retry	30	Time in seconds to wait before attempting to reconnect to a server that is down.
request-queue-size	10	The number of requests that can be queued if the connection pool is full before requests are rejected with a 503.
session-cookie-names	JSESSIONID	Comma-separated list of session cookie names. Generally this will just be JSESSIONID.

Table A.83. host Attributes

Attribute	Default	Description
enable-http2	false	If true , then the proxy will attempt to use HTTP/2 to connect to the back end. If it is not supported, it will fall back to HTTP/1.1.
instance-id		The instance ID, or JVM route, that will be used to enable sticky sessions.
outbound-socket-binding		Outbound socket binding for this host.
path	/	Optional path if host is using non root resource.
scheme	http	The kind of scheme that is used.
security-realm		The security realm that provides the SSL configuration for the connection to the host.
ssl-context		Reference to the SSLContext to be used by this handler.

Server Attributes

The server component has the following structure:

- **server**
 - **ajp-listener**
 - **host**
 - **filter-ref**
 - **location**
 - **filter-ref**

- **setting**
 - [access-log](#)
 - [console-access-log](#)
 - [http-invoker](#)
 - [single-sign-on](#)
- [http-listener](#)
- [https-listener](#)

server Attributes

Table A.84. server Attributes

Attribute	Default	Description
default-host	default-host	The server's default virtual host.
servlet-container	default	The server's default servlet container.

ajp-listener Attributes

Table A.85. ajp-listener Attributes

Attribute	Default	Description
allow-encoded-slash	false	If a request comes in with encoded characters, for example %2F , whether these will be decoded.
allow-equals-in-cookie-value	false	Whether to allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.
allow-unescaped-characters-in-url	false	Whether to allow non-escaped characters in a URL. If set to true , the listener processes any URL containing non-escaped, non-ASCII characters. If set to false , the listener rejects any URL containing non-escaped, non-ASCII characters with an HTTP Bad Request 400 response code.
always-set-keep-alive	true	Whether a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.
buffer-pipelined-data	false	Whether to buffer pipelined requests.

Attribute	Default	Description
buffer-pool	default	The AJP listener's buffer pool.
decode-url	true	If this is true then the parser will decode the URL and query parameters using the selected character encoding, defaulting to UTF-8. If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.
disallowed-methods	["TRACE"]	A comma-separated list of HTTP methods that are not allowed.
enabled	true	If the listener is enabled. <i>Deprecated: Enabled attributes can cause problems in enforcement of configuration consistency.</i>
max-ajp-packet-size	8192	The maximum supported size of AJP packets. If this is modified it has to be increased on the load balancer and the back-end server.
max-buffered-request-size	16384	Maximum size of a buffered request, in bytes. Requests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.
max-connections		The maximum number of concurrent connections. If no value is set in the server configuration, the limit for the number of concurrent connections is Integer.MAX_VALUE .
max-cookies	200	The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.
max-header-size	1048576	The maximum size in bytes of an HTTP request header.
max-headers	200	The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.
max-parameters	1000	The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative. For example, you can potentially have max parameters * 2 total parameters.
max-post-size	10485760	The maximum size of a post that will be accepted

Attribute	Default	Description
no-request-timeout	60000	The length of time in milliseconds that the connection can be idle before it is closed by the container.
read-timeout		Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a ReadTimeoutException .
receive-buffer		The receive buffer size.
record-request-start-time	false	Whether to record the request start time, to allow for request time to be logged. This has a small but measurable performance impact.
redirect-socket		If this listener is supporting non-SSL requests, and a request is received for which a matching requires SSL transport, whether to automatically redirect the request to the socket binding port specified here.
request-parse-timeout		The maximum amount of time in milliseconds that can be spent parsing the request.
resolve-peer-address	false	Enables host DNS lookup.
scheme		The listener scheme, can be HTTP or HTTPS. By default the scheme will be taken from the incoming AJP request.
secure	false	If this is true , then requests that originate from this listener are marked as secure, even if the request is not using HTTPS.
send-buffer		The send buffer size.
socket-binding		The AJP listener's socket binding.
tcp-backlog		Configure a server with the specified backlog.
tcp-keep-alive		Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.
url-charset	UTF-8	URL charset.
worker	default	The listener's XNIO worker.

Attribute	Default	Description
write-timeout		Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a WriteTimeoutException .

host Attributes

Table A.86. host Attributes

Attribute	Default	Description
alias		Comma-separated list of aliases for the host.
default-response-code	404	If set, this will be response code sent back in case requested context does not exist on server.
default-web-module	ROOT.war	Default web module.
disable-console-redirect	false	If set to true , /console redirect will not be enabled for this host.
queue-requests-on-start	true	If set to true , requests should be queued on start for this host. If set to false , the default response code is returned instead.

filter-ref Attributes

Table A.87. filter-ref Attributes

Attribute	Default	Description
predicate		Predicates provide a simple way of making a true/false decision based on an exchange. Many handlers have a requirement that they be applied conditionally, and predicates provide a general way to specify a condition.
priority	1	Defines filter order. A lower number instructs the server to be included earlier in the handler chain than others above the same context. Values range from 1 , indicating the filter will be handled first, to 2147483647 , resulting in the filter being handled last.

location Attributes

Table A.88. location Attributes

Attribute	Default	Description
handler		Default handler for this location.

filter-ref Attributes

Table A.89. filter-ref Attributes

Attribute	Default	Description
predicate		Predicates provide a simple way of making a true/false decision based on an exchange. Many handlers have a requirement that they be applied conditionally, and predicates provide a general way to specify a condition.
priority	1	Defines filter order. It should be set to 1 or more. A higher number instructs the server to be included earlier in the handler chain than others under the same context.

access-log Attributes



NOTE

When using the management CLI to manage the **access-log** element, it is available under **settings** in the **host** element. For example:

```
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:add
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:read-resource
```

Table A.90. access-log Attributes

Attribute	Default	Description
directory	\${boss.server.log.dir}	The directory in which to save logs.
extended	false	Whether the log uses the extended log file format.

Attribute	Default	Description
pattern	common	<p>The access log pattern. For details about the options available for this attribute, see Provided Undertow Handlers in the JBoss EAP <i>Development Guide</i>.</p> <div>  <div> <p>NOTE</p> <p>If you set the pattern to print the time taken to process the request, you must also enable the record-request-start-time attribute on the appropriate listeners; otherwise the time will not be recorded properly in the access log. For example:</p> <pre>/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=record-request-start-time,value=true)</pre> </div> </div>
predicate		Predicate that determines whether the request should be logged.
prefix	access_log.	Prefix for the log file name.
relative-to		The directory the path is relative to.
rotate	true	Whether to rotate the access log every day.
suffix	log	Suffix for the log file name.
use-server-log	false	Whether the log should be written to the server log, rather than a separate file.
worker	default	Name of the worker to use for logging.

console-access-log Attributes

Table A.91. console-access-log attributes

Attribute	Default	Description
attributes	{remote-host={},remote-user={},date-time={},request-line={},response-code={},bytes-sent={}}	Specifies log data to include in the console access log output, or customizations to default data.
include-host-name	false	Specifies whether to include the host name in the JSON structured output. If set to true the key in the structured data is "hostName" and the value is the name of the host for which the console-access-log is configured.
metadata		Specifies custom metadata to include in console access log output.
predicate		Predicate that determines whether the request should be logged.
worker	default	Name of the worker to use for logging.

http-invoker Attributes

Table A.92. http-invoker Attributes

Attribute	Default	Description
http-authentication-factory		The HTTP authentication factory to use for authentication.
path	wildfly-services	The path that the services are installed under.
security-realm		The legacy security realm to use for authentication.

single-sign-on Attributes

**NOTE**

When using the management CLI to manage the **single-sign-on** element, it is available under **settings** in the **host** element. For example:

```
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:add
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:read-resource
```

**IMPORTANT**

While distributed single sign-on is no different from an application perspective from previous versions of JBoss EAP, in JBoss EAP 7 the caching and distribution of authentication information is handled differently. For JBoss EAP 7, when running the *ha* profile, by default each host will have its own Infinispan cache which will store the relevant session and SSO cookie information. This cache is based on the default cache of the web cache container. JBoss EAP will also handle propagating information between all hosts' individual caches.

Table A.93. single-sign-on Attributes

Attribute	Default	Description
cookie-name	JSESSIONIDS SO	Name of the cookie.
domain		The cookie domain that will be used.
http-only	false	Set cookie httpOnly attribute.
path	/	Cookie path.
secure	false	Set cookie secure attribute.

http-listener Attributes**Table A.94. http-listener Attributes**

Attribute	Default	Description
allow-encoded-slash	false	If a request comes in with encoded characters, for example %2F , whether these will be decoded.
allow-equals-in-cookie-value	false	Whether to allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.

Attribute	Default	Description
allow-unescaped-characters-in-url	false	Whether to allow non-escaped characters in a URL. If set to true , the listener processes any URL containing non-escaped, non-ASCII characters. If set to false , the listener rejects any URL containing non-escaped, non-ASCII characters with an HTTP Bad Request 400 response code.
always-set-keep-alive	true	Whether a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.
buffer-pipelined-data	false	Whether to buffer pipelined requests.
buffer-pool	default	The listener's buffer pool.
certificate-forwarding	false	Whether certificate forwarding should be enabled. If this is enabled then the listener will take the certificate from the SSL_CLIENT_CERT attribute. This should only be enabled if behind a proxy, and the proxy is configured to always set these headers.
decode-url	true	Whether the parser will decode the URL and query parameters using the selected character encoding, defaulting to UTF-8. If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.
disallowed-methods	["TRACE"]	A comma-separated list of HTTP methods that are not allowed.
enable-http2	false	Whether to enable HTTP/2 support for this listener.
enabled	true	Whether the listener is enabled. <i>Deprecated: Enabled attributes can cause problems in enforcement of configuration consistency.</i>
http2-enable-push	true	Whether server push is enabled for this connection.
http2-header-table-size	4096	The size, in bytes, of the header table used for HPACK compression. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.
http2-initial-window-size	65535	The flow control window size, in bytes, that controls how quickly the client can send data to the server.

Attribute	Default	Description
http2-max-concurrent-streams		The maximum number of HTTP/2 streams that can be active at any time on a single connection.
http2-max-frame-size	16384	The maximum HTTP/2 frame size, in bytes.
http2-max-header-list-size		The maximum size of request headers the server is prepared to accept.
max-buffered-request-size	16384	Maximum size of a buffered request, in bytes. Requests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.
max-connections		The maximum number of concurrent connections. If no value is set in the server configuration, the limit for the number of concurrent connections is Integer.MAX_VALUE .
max-cookies	200	The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.
max-header-size	1048576	The maximum size in bytes of an HTTP request header.
max-headers	200	The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.
max-parameters	1000	The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative. For example, you can potentially have max parameters * 2 total parameters).
max-post-size	10485760	The maximum size of a post that will be accepted.
no-request-timeout	60000	The length of time in milliseconds that the connection can be idle before it is closed by the container.
proxy-address-forwarding	false	Whether to enable x-forwarded-host and similar headers and set a remote IP address and host name.

Attribute	Default	Description
proxy-protocol	false	Whether to use the PROXY protocol to transport connection information. If set to true , the listener uses the PROXY protocol Version 1, as defined by The PROXY protocol Versions 1 & 2 specification. This option must only be enabled for listeners that are behind a load balancer that supports the same protocol.
read-timeout		Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a ReadTimeoutException .
receive-buffer		The receive buffer size.
record-request-start-time	false	Whether to record the request start time, to allow for request time to be logged. This has a small but measurable performance impact.
redirect-socket		If this listener is supporting non-SSL requests, and a request is received for which a matching requires SSL transport, whether to automatically redirect the request to the socket binding port specified here.
request-parse-timeout		The maximum amount of time in milliseconds that can be spent parsing the request.
require-host-http11	false	It requires all HTTP/1.1 requests to have a Host header. If the request does not include this header it will be rejected with a 403 error.
resolve-peer-address	false	Enables host DNS lookup.
secure	false	If this is true , requests that originate from this listener are marked as secure, even if the request is not using HTTPS.
send-buffer		The send buffer size.
socket-binding		The listener's socket binding
tcp-backlog		Configure a server with the specified backlog.
tcp-keep-alive		Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.
url-charset	UTF-8	URL charset.

Attribute	Default	Description
worker	default	The listener's XNIO worker.
write-timeout		Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a WriteTimeoutException .

https-listener Attributes

Table A.95. https-listener Attributes

Attribute	Default	Description
allow-encoded-slash	false	If a request comes in with encoded characters, for example %2F , whether these will be decoded.
allow-equals-in-cookie-value	false	Whether to allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.
allow-unescaped-characters-in-url	false	Whether to allow non-escaped characters in a URL. If set to true , the listener processes any URL containing non-escaped, non-ASCII characters. If set to false , the listener rejects any URL containing non-escaped, non-ASCII characters with an HTTP Bad Request 400 response code.
always-set-keep-alive	true	Whether a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.
buffer-pipelined-data	false	Whether to buffer pipelined requests.
buffer-pool	default	The listener's buffer pool.
certificate-forwarding	false	Whether certificate forwarding should be enabled or not. If this is enabled then the listener will take the certificate from the SSL_CLIENT_CERT attribute. This should only be enabled if behind a proxy, and the proxy is configured to always set these headers.
decode-url	true	Whether the parser will decode the URL and query parameters using the selected character encoding, defaulting to UTF-8. If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.

Attribute	Default	Description
disallowed-methods	["TRACE"]	A comma-separated list of HTTP methods that are not allowed.
enable-http2	false	Enables HTTP/2 support for this listener.
enable-spdy	false	Enables SPDY support for this listener. <i>Deprecated: SPDY has been replaced by HTTP/2.</i>
enabled	true	If the listener is enabled. <i>Deprecated: Enabled attributes can cause problems in enforcement of configuration consistency.</i>
enabled-cipher-suites		Configures Enabled SSL ciphers. <i>Deprecated: Where an SSLContext is referenced it should be configured with the cipher suites to be supported.</i>
enabled-protocols		Configures SSL protocols. <i>Deprecated: Where an SSLContext is referenced it should be configured with the cipher suites to be supported.</i>
http2-enable-push	true	If server push is enabled for this connection.
http2-header-table-size	4096	The size, in bytes, of the header table used for HPACK compression. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.
http2-initial-window-size	65535	The flow control window size, in bytes, that controls how quickly the client can send data to the server.
http2-max-concurrent-streams		The maximum number of HTTP/2 streams that can be active at any time on a single connection.
http2-max-frame-size	16384	The maximum HTTP/2 frame size, in bytes.
http2-max-header-list-size		The maximum size of request headers the server is prepared to accept.
max-buffered-request-size	16384	Maximum size of a buffered request, in bytes. Requests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.

Attribute	Default	Description
max-connections		The maximum number of concurrent connections. If no value is set in the server configuration, the limit for the number of concurrent connections is Integer.MAX_VALUE .
max-cookies	100	The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.
max-header-size	1048576	The maximum size in bytes of an HTTP request header.
max-headers	200	The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.
max-parameters	1000	The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative. For example, you can potentially have max parameters * 2 total parameters.
max-post-size	10485760	The maximum size of a post that will be accepted.
no-request-timeout	60000	The length of time in milliseconds that the connection can be idle before it is closed by the container.
proxy-address-forwarding	false	Enables handling of x-forwarded-host header, and other x-forwarded-* headers, and uses this header information to set the remote address. This should only be used behind a trusted proxy that sets these headers otherwise a remote user can spoof their IP address.
proxy-protocol	false	Whether to use the PROXY protocol to transport connection information. If set to true , the listener uses the PROXY protocol Version 1, as defined by The PROXY protocol Versions 1 & 2 specification. This option must only be enabled for listeners that are behind a load balancer that supports the same protocol.
read-timeout		Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a ReadTimeoutException .
receive-buffer		The receive buffer size.

Attribute	Default	Description
record-request-start-time	false	Whether to record the request start time, to allow for request time to be logged. This has a small but measurable performance impact.
request-parse-timeout		The maximum amount of time in milliseconds that can be spent parsing the request.
require-host-http11	false	Require that all HTTP/1.1 requests have a 'Host' header. If the request does not include this header it will be rejected with a 403.
resolve-peer-address	false	Enables host DNS lookup.
secure	false	If this is true then requests that originate from this listener are marked as secure, even if the request is not using HTTPS.
security-realm		The listener's security realm. <i>Deprecated: Use the ssl-context attribute to reference a configured SSLContext directly.</i>
send-buffer		The send buffer size.
socket-binding		The listener's socket binding.
ssl-context		Reference to the SSLContext to be used by this listener.
ssl-session-cache-size		The maximum number of active SSL sessions. <i>Deprecated: This can now be configured on the Elytron security context.</i>
ssl-session-timeout		The timeout for SSL sessions, in seconds. <i>Deprecated: This can now be configured on the Elytron security context.</i>
tcp-backlog		Configure a server with the specified backlog.
tcp-keep-alive		Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.
url-charset	UTF-8	URL charset.
verify-client	NOT_REQUESTED	The desired SSL client authentication mode for SSL channels. <i>Deprecated: Where an SSLContext is referenced it should be configured directly for the required mode of client verification.</i>

Attribute	Default	Description
worker	default	The listener's XNIO worker.
write-timeout		Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a WriteTimeoutException .

A.28. UNDERTOW SUBSYSTEM STATISTICS

Table A.96. ajp-listener Statistics

Name	Description
bytes-received	The number of bytes that have been received by this listener.
bytes-sent	The number of bytes that have been sent out on this listener.
error-count	The number of 500 responses that have been sent by this listener.
max-processing-time	The maximum processing time taken by a request on this listener.
processing-time	The total processing time of all requests handed by this listener.
request-count	The number of requests this listener has served.

Table A.97. http-listener Statistics

Name	Description
bytes-received	The number of bytes that have been received by this listener.
bytes-sent	The number of bytes that have been sent out on this listener.
error-count	The number of 500 responses that have been sent by this listener.
max-processing-time	The maximum processing time taken by a request on this listener.
processing-time	The total processing time of all requests handed by this listener.
request-count	The number of requests this listener has served.

Table A.98. https-listener Statistics

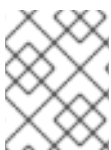
Name	Description
bytes-received	The number of bytes that have been received by this listener.
bytes-sent	The number of bytes that have been sent out on this listener.
error-count	The number of 500 responses that have been sent by this listener.
max-processing-time	The maximum processing time taken by a request on this listener.
processing-time	The total processing time of all requests handed by this listener.
request-count	The number of requests this listener has served.

A.29. DEFAULT BEHAVIOR OF HTTP METHODS

Compared to the **web** subsystem in previous JBoss EAP releases, the **undertow** subsystem in JBoss EAP 7.3 has different default behaviors of HTTP methods. The following table outlines the default behaviors in JBoss EAP 7.3.

Table A.99. HTTP Method Default Behavior

HTTP Method	JSP	Static HTML	Static HTML by File Handler
GET	OK	OK	OK
POST	OK	NOT_ALLOWED	OK
HEAD	OK	OK	OK
PUT	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED
TRACE	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED
DELETE	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED
OPTIONS	NOT_ALLOWED	OK	NOT_ALLOWED



NOTE

For servlets, the default behavior depends on its implementation, except for the **TRACE** method, which has a default behavior of **NOT_ALLOWED**.

A.30. REMOTING SUBSYSTEM ATTRIBUTES

**NOTE**

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-remoting_4_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.100. remoting Attributes

Attribute	Default	Description
worker-read-threads	1	The number of read threads to create for the remoting worker.
worker-task-core-threads	4	The number of core threads for the remoting worker task thread pool.
worker-task-keepalive	60	The number of milliseconds to keep non-core remoting worker task threads alive.
worker-task-limit	16384	The maximum number of remoting worker tasks to allow before rejecting.
worker-task-max-threads	16	The maximum number of threads for the remoting worker task thread pool.
worker-write-threads	1	The number of write threads to create for the remoting worker.

**IMPORTANT**

The above attributes of the **remoting** element are deprecated. These attributes should now be configured using the **io** subsystem.

Table A.101. endpoint Attributes

Attribute	Default	Description
auth-realm		The authentication realm to use if no authentication CallbackHandler is specified.
authentication-retries	3	Specify the number of times a client is allowed to retry authentication before closing the connection.
authorize-id		The SASL authorization ID. Used as authentication user name to use if no authentication CallbackHandler is specified and the selected SASL mechanism demands a user name.
buffer-region-size		The size of allocated buffer regions.

Attribute	Default	Description
heartbeat-interval	2147483647	The interval to use for connection heartbeat, in milliseconds. If the connection is idle in the outbound direction for this amount of time, a ping message will be sent, which will trigger a corresponding reply message.
max-inbound-channels	40	The maximum number of concurrent inbound messages on a channel.
max-inbound-message-size	9223372036854775807	The maximum inbound message size to be allowed. Messages exceeding this size will cause an exception to be thrown on the reading side as well as the writing side.
max-inbound-messages	80	The maximum number of inbound channels to support for a connection.
max-outbound-channels	40	The maximum number of concurrent outbound messages on a channel.
max-outbound-message-size	9223372036854775807	The maximum outbound message size to send. No messages larger than this will be transmitted; attempting to do so will cause an exception on the writing side.
max-outbound-messages	65535	The maximum number of outbound channels to support for a connection.
receive-buffer-size	8192	The size of the largest buffer that this endpoint will accept over a connection.
receive-window-size	131072	The maximum window size of the receive direction for connection channels, in bytes.
sasl-protocol	remote	When a SaslServer or SaslClient is created, the protocol specified by default is remote . This attribute can be used to override this protocol.
send-buffer-size	8192	The size of the largest buffer that this endpoint will transmit over a connection.
server-name		The server side of the connection passes its name to the client in the initial greeting, by default the name is automatically discovered from the local address of the connection or it can be overridden using this.
transmit-window-size	131072	The maximum window size of the transmit direction for connection channels, in bytes.

Attribute	Default	Description
worker	default	Worker to use

**NOTE**

When using the management CLI to update the **endpoint** element, it is available under **configuration** in the **remoting** element. For example:
/subsystem=remoting/configuration=endpoint/.

Connector Attributes

The connector component has the following structure:

- `connector`
 - `property`
 - `security`
 - `sasl`
 - `property`
 - `sasl-policy`
 - `policy`

Table A.102. connector Attributes

Attribute	Default	Description
authentication-provider		The authentication-provider element contains the name of the authentication provider to use for incoming connections.
sasl-authentication-factory		Reference to the SASL authentication factory to secure this connector.
sasl-protocol	remote	The protocol to pass into the SASL mechanisms used for authentication.
security-realm		The associated security realm to use for authentication for this connector.
server-name		The server name to send in the initial message exchange and for SASL based authentication.
socket-binding		The name (or names) of the socket binding(s) to attach to.

Attribute	Default	Description
ssl-context		Reference to the SSL context to use for this connector.

Table A.103. property Attributes

Attribute	Default	Description
value		The property value.

Security Attributes

The **security** component allows you to configure the security for the connector, but contains no direct configuration attributes. It can be configured using its nested components, such as [sasl](#).

Table A.104. sasl Attributes

Attribute	Default	Description
include-mechanisms		The optional nested include-mechanisms element contains a whitelist of allowed SASL mechanism names. No mechanisms will be allowed which are not present in this list.
qop		<p>The optional nested qop element contains a comma-separated list of quality-of-protection values, in decreasing order of preference.</p> <p>Quality-of-protection values for this list are:</p> <ul style="list-style-type: none"> ● auth: authentication only ● auth-int: authentication, plus integrity protection ● auth-conf: authentication, plus integrity protection and confidentiality protection
reuse-session	false	The optional nested reuse-session boolean element specifies whether or not the server should attempt to reuse previously authenticated session information. The mechanism may or may not support such reuse, and other factors may also prevent it.
server-auth	false	The optional nested server-auth boolean element specifies whether the server should authenticate to the client. Not all mechanisms may support this setting.

Attribute	Default	Description
strength		<p>The optional nested strength element contains a comma-separated list of cipher strength values, in decreasing order of preference.</p> <p>Cipher strength values for this list are:</p> <ul style="list-style-type: none"> • high • medium • low

sasl-policy Attributes

The **sasl-policy** component allows you to specify an optional policy to use to narrow down the available set of mechanisms, but contains no direct configuration attributes. It can be configured using its nested components, such as [policy](#).

Table A.105. policy Attributes

Attribute	Default	Description
forward-secrecy	true	The optional nested forward-secrecy element contains a boolean value which specifies whether mechanisms that implement forward secrecy between sessions are required. Forward secrecy means that breaking into one session will not automatically provide information for breaking into future sessions.
no-active	true	The optional nested no-active element contains a boolean value which specifies whether mechanisms susceptible to active (non-dictionary) attacks are not permitted. false to permit, true to deny.
no-anonymous	true	The optional nested no-anonymous element contains a boolean value which specifies whether mechanisms that accept anonymous login are permitted. false to permit, true to deny.
no-dictionary	true	The optional nested no-dictionary element contains a boolean value which specifies whether mechanisms susceptible to passive dictionary attacks are permitted. false to permit, true to deny.

Attribute	Default	Description
no-plain-text	true	The optional nested no-plain-text element contains a boolean value which specifies whether mechanisms susceptible to simple plain passive attacks (for example, PLAIN) are not permitted. false to permit, true to deny.
pass-credentials	true	The optional nested pass-credentials element contains a boolean value which specifies whether mechanisms that pass client credentials are required.

HTTP Connector Attributes

The http-connector component has the following structure:

- [http-connector](#)
 - [property](#) (same as connector)
 - [security](#) (same as connector)
 - [sasl](#) (same as connector)
 - [property](#) (same as connector)
 - [sasl-policy](#) (same as connector)
 - [policy](#) (same as connector)

Table A.106. http-connector Attributes

Attribute	Default	Description
authentication-provider		The authentication-provider element contains the name of the authentication provider to use for incoming connections.
connector-ref		The name (or names) of a connector in the undertow subsystem to connect to.
sasl-authentication-factory		Reference to the SASL authentication factory to secure this connector.
sasl-protocol	remote	The protocol to pass into the SASL mechanisms used for authentication.
security-realm		The associated security realm to use for authentication for this connector.
server-name		The server name to send in the initial message exchange and for SASL based authentication.

Outbound Connection Attributes

The **outbound-connection** component has the following structure:

- [outbound-connection](#)
 - [property](#)

Table A.107. outbound-connection Attributes

Attribute	Default	Description
uri		The connection URI for the outbound connection.

Table A.108. property Attributes

Attribute	Default	Description
value		The property value.



NOTE

The above **property** attributes are related to the XNIO Options that will be used during the connection creation.

Remote Outbound Connection

The **remote-outbound-connection** component has the following structure:

- [remote-outbound-connection](#)
 - [property \(same as outbound-connection\)](#)

Table A.109. remote-outbound-connection Attributes

Attribute	Default	Description
authentication-context		Reference to the authentication context instance containing the configuration for outbound connections.
outbound-socket-binding-ref		Name of the outbound-socket-binding which will be used to determine the destination address and port for the connection.
protocol	http-remoting	The protocol to use for the remote connection. Defaults to http-remoting . <i>Deprecated: Outbound security settings should be migrated to an authentication-context definition.</i>

Attribute	Default	Description
security-realm		Reference to the security realm to use to obtain the password and SSL configuration. <i>Deprecated: Outbound security settings should be migrated to an authentication-context definition.</i>
username		The user name to use when authenticating against the remote server. <i>Deprecated: Outbound security settings should be migrated to an authentication-context definition.</i>

Local Outbound Connection Attributes

The **local-outbound-connection** component has the following structure:

- [local-outbound-connection](#)
 - [property](#) (same as [outbound-connection](#))

Table A.110. local-outbound-connection Attributes

Attribute	Default	Description
outbound-socket-binding-ref		Name of the outbound-socket-binding which will be used to determine the destination address and port for the connection.

A.31. IO SUBSYSTEM ATTRIBUTES



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-io_2_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.111. worker Attributes

Attribute	Default	Description
io-threads		The number of I/O threads to create for the worker. If not specified, the number of threads is set to the number of CPUs × 2.
stack-size	0	The stack size, in bytes, to attempt to use for worker threads.

Attribute	Default	Description
task-keepalive	60000	The number of milliseconds to keep non-core task threads alive.
task-core-threads	2	The number of threads for the core task thread pool.
task-max-threads		The maximum number of threads for the worker task thread pool. If not specified, the maximum number of threads is set to the number of CPUs × 16, taking the MaxFileDescriptorCount JMX property, if set, into account.

Table A.112. buffer-pool Attributes

Attribute	Default	Description
buffer-size		<p>The size, in bytes, of each buffer slice. If not specified, the size is set based on the available RAM of your system:</p> <ul style="list-style-type: none"> ● 512 bytes for less than 64 MB RAM ● 1024 bytes (1 KB) for 64 MB - 128 MB RAM ● 16384 bytes (16 KB) for more than 128 MB RAM <p>For performance tuning advice on this attribute, see Configuring Buffer Pools in the JBoss EAP <i>Performance Tuning Guide</i>.</p>
buffers-per-slice		<p>How many slices, or sections, to divide the larger buffer into. This can be more memory efficient than allocating many separate buffers. If not specified, the number of slices is set based on the available RAM of your system:</p> <ul style="list-style-type: none"> ● 10 for less than 128 MB RAM ● 20 for more than 128 MB RAM
direct-buffers		Whether the buffer pool uses direct buffers, which are faster in many cases with NIO. Note that some platforms do not support direct buffers.

A.32. JAKARTA SERVER FACES MODULE TEMPLATES

The following are example templates used for the various Jakarta Server Faces modules required when installing a different Jakarta Server Faces version for JBoss EAP. See [Installing a Jakarta Server Faces Implementation](#) for full instructions.

Example: Mojarra Jakarta Server Faces Implementation JAR module.xml



NOTE

Be sure to use the appropriate values for the following replaceable variables in the template:

- ***IMPL_NAME***
- ***VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="com.sun.jsf-impl:IMPL_NAME-VERSION">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <dependencies>
    <module name="javax.faces.api:IMPL_NAME-VERSION"/>
    <module name="javaee.api"/>
    <module name="javax.servlet.jstl.api"/>
    <module name="org.apache.xerces" services="import"/>
    <module name="org.apache.xalan" services="import"/>
    <module name="org.jboss.weld.core"/>
    <module name="org.jboss.weld.spi"/>
    <module name="javax.xml.rpc.api"/>
    <module name="javax.rmi.api"/>
    <module name="org.omg.api"/>
  </dependencies>

  <resources>
    <resource-root path="impl-VERSION.jar"/>
  </resources>
</module>
```

Example: MyFaces Jakarta Server Faces Implementation JAR module.xml



NOTE

Be sure to use the appropriate values for the following replaceable variables in the template:

- ***IMPL_NAME***
- ***VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="com.sun.jsf-impl:IMPL_NAME-VERSION">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>
```

```

<dependencies>
  <module name="javax.faces.api:IMPL_NAME-VERSION">
    <imports>
      <include path="META-INF/**"/>
    </imports>
  </module>
  <module name="javaee.api"/>
  <module name="javax.servlet.jstl.api"/>
  <module name="org.apache.xerces" services="import"/>
  <module name="org.apache.xalan" services="import"/>

  <!-- extra dependencies for MyFaces -->
  <module name="org.apache.commons.collections"/>
  <module name="org.apache.commons.codec"/>
  <module name="org.apache.commons.beanutils"/>
  <module name="org.apache.commons.digester"/>

  <!-- extra dependencies for MyFaces 1.1 -->
  <module name="org.apache.commons.logging"/>
  <module name="org.apache.commons.el"/>
  <module name="org.apache.commons.lang"/> -->
  <module name="javax.xml.rpc.api"/>
  <module name="javax.rmi.api"/>
  <module name="org.omg.api"/>
</dependencies>

<resources>
  <resource-root path="IMPL_NAME-impl-VERSION.jar"/>
</resources>
</module>

```

Example: Mojarra Jakarta Server Faces API JAR module.xml



NOTE

Be sure to use the appropriate values for the following replaceable variables in the template:

- **IMPL_NAME**
- **VERSION**

```

<module xmlns="urn:jboss:module:1.8" name="javax.faces.api:IMPL_NAME-VERSION">
  <dependencies>
    <module name="com.sun.jsf-impl:IMPL_NAME-VERSION"/>
    <module name="javax.enterprise.api" export="true"/>
    <module name="javax.servlet.api" export="true"/>
    <module name="javax.servlet.jsp.api" export="true"/>
    <module name="javax.servlet.jstl.api" export="true"/>
    <module name="javax.validation.api" export="true"/>
    <module name="org.glassfish.javax.el" export="true"/>
    <module name="javax.api"/>
    <module name="javax.websocket.api"/>
  </dependencies>

```

```

<resources>
  <resource-root path="jsf-api-VERSION.jar"/>
</resources>
</module>

```

Example: MyFaces Jakarta Server Faces API JAR module.xml



NOTE

Be sure to use the appropriate values for the following replaceable variables in the template:

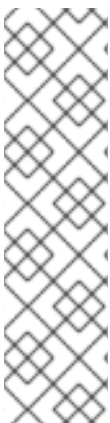
- ***IMPL_NAME***
- ***VERSION***

```

<module xmlns="urn:jboss:module:1.8" name="javax.faces.api:IMPL_NAME-VERSION

```

Example: Mojarra Jakarta Server Faces Injection JAR module.xml



NOTE

Be sure to use the appropriate values for the following replaceable variables in the template:

- ***IMPL_NAME***
- ***VERSION***
- ***INJECTION_VERSION***
- ***WELD_VERSION***

```

<module xmlns="urn:jboss:module:1.8" name="org.jboss.as.jsf-injection:IMPL_NAME-VERSION

```

```

    <property name="jboss.api" value="private"/>
  </properties>

  <resources>
    <resource-root path="wildfly-jsf-injection-INJECTION_VERSION.jar"/>
    <resource-root path="weld-core-jsf-WELD_VERSION.jar"/>
  </resources>

  <dependencies>
    <module name="com.sun.jsf-impl:IMPL_NAME-VERSION"/>
    <module name="java.naming"/>
    <module name="java.desktop"/>
    <module name="org.jboss.as.jsf"/>
    <module name="org.jboss.as.web-common"/>
    <module name="javax.servlet.api"/>
    <module name="org.jboss.as.ee"/>
    <module name="org.jboss.as.jsf"/>
    <module name="javax.enterprise.api"/>
    <module name="org.jboss.logging"/>
    <module name="org.jboss.weld.core"/>
    <module name="org.jboss.weld.api"/>

    <module name="javax.faces.api:IMPL_NAME-VERSION"/>
  </dependencies>
</module>

```

Example: MyFaces Jakarta Server Faces Injection JAR module.xml



NOTE

Be sure to use the appropriate values for the following replaceable variables in the template:

- ***IMPL_NAME***
- ***VERSION***
- ***INJECTION_VERSION***
- ***WELD_VERSION***

```

<module xmlns="urn:jboss:module:1.8" name="org.jboss.as.jsf-injection:IMPL_NAME-VERSION">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <resources>
    <resource-root path="wildfly-jsf-injection-INJECTION_VERSION.jar"/>
    <resource-root path="weld-jsf-WELD_VERSION.jar"/>
  </resources>

  <dependencies>
    <module name="com.sun.jsf-impl:IMPL_NAME-VERSION"/>
    <module name="javax.api"/>
    <module name="org.jboss.as.web-common"/>

```



```

<module name="javax.servlet.api"/>
<module name="org.jboss.as.jsf"/>
<module name="org.jboss.as.ee"/>
<module name="org.jboss.as.jsf"/>
<module name="javax.enterprise.api"/>
<module name="org.jboss.logging"/>
<module name="org.jboss.weld.core"/>
<module name="org.jboss.weld.api"/>
<module name="org.wildfly.security.elytron"/>

<module name="javax.faces.api:IMPL_NAME-VERSION"/>
</dependencies>
</module>

```

Example: MyFaces commons-digester JAR module.xml



NOTE

Be sure to use the appropriate value for the **VERSION** replaceable variable in the template.

```

<module xmlns="urn:jboss:module:1.5" name="org.apache.commons.digester">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <resources>
    <resource-root path="commons-digester-VERSION.jar"/>
  </resources>

  <dependencies>
    <module name="javax.api"/>
    <module name="org.apache.commons.collections"/>
    <module name="org.apache.commons.logging"/>
    <module name="org.apache.commons.beanutils"/>
  </dependencies>
</module>

```

A.33. JGROUPS SUBSYSTEM ATTRIBUTES

See the tables below for the attributes of the various elements of the **jgroups** subsystem.

- [Main Attributes](#)
- [Channel Attributes](#)
- [Stack Attributes](#)



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at **EAP_HOME/docs/schema/jboss-as-jgroups_5_0.xsd** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.113. Main jgroups Attributes

Attribute	Default	Description
default-channel	ee	The default JGroups channel.
default-stack		The default JGroups protocol stack.

Channel Attributes

The channel element has the following structure:

- **channel**
 - **fork**
 - **protocol**
 - **protocol**

channel Attributes

Table A.114. channel Attributes

Attribute	Default	Description
cluster		The cluster name of the JGroups channel. If undefined, the name of the channel will be used.
module	org.wildfly.clustering.server	The module from which to load channel services.
stack		The protocol stack of the JGroups channel.
statistics-enabled	false	Whether statistics are enabled.
stats-enabled	false	Whether statistics are enabled. <i>Deprecated: Use the statistics-enabled attribute instead.</i>

Stack Attributes

The stack element has the following structure:

- **stack**
 - **protocol**
 - **relay**
 - **remote-site**
 - **transport**
 - **thread-pool**

stack Attributes**Table A.115. stack Attributes**

Attribute	Default	Description
statistics-enabled	false	Indicates whether or not all protocols in the stack will collect statistics.

protocol Attributes

For a list of commonly used protocols, see the [JGroups Protocols](#) section.

Table A.116. protocol Attributes

Attribute	Default	Description
module	org.jgroups	The module with which to resolve the protocol type.
properties		Properties of this protocol.
statistics-enabled	false	Indicates whether or not this protocol will collect statistics, overriding the stack configuration.

relay Attributes**Table A.117. relay Attributes**

Attribute	Default	Description
module	org.jgroups	The module with which to resolve the protocol type.
properties		Properties of this protocol.
site		The name of the local site.
statistics-enabled	false	Indicates whether or not this protocol will collect statistics, overriding the stack configuration.

remote-site Attributes**Table A.118. remote-site Attributes**

Attribute	Default	Description
channel		The name of the bridge channel used to communicate with this remote site.

Attribute	Default	Description
cluster		The cluster name of the bridge channel to this remote site. <i>Deprecated: Use an explicitly defined channel instead.</i>
stack		The stack from which to create a bridge to this remote site. <i>Deprecated: Use an explicitly defined channel instead.</i>

transport Attributes

Table A.119. transport Attributes

Attribute	Default	Description
default-executor		The thread pool executor to handle incoming messages. <i>Deprecated: Configure the predefined default thread pool instead.</i>
diagnostics-socket-binding		The diagnostics socket binding specification for this protocol layer, used to specify IP interfaces and ports for communication.
machine		Machine, or host, identifier for this node. Used by Infinispan's topology-aware consistent hash.
module	org.jgroups	Module with which to resolve the protocol type.
oob-executor		The thread pool executor to handle incoming out-of-band messages. <i>Deprecated: Configure the predefined oob thread pool instead.</i>
properties		Properties of this transport.
rack		Rack, such as the server rack, identifier for this node. Used by Infinispan's topology-aware consistent hash.
shared	false	If true , the underlying transport is shared by all channels using this stack. <i>Deprecated: Configure a fork of the channel instead.</i>
site		Site, such as the data center, identifier for this node. Used by Infinispan's topology-aware consistent hash.
socket-binding		The socket binding specification for this protocol layer, used to specify IP interfaces and ports for communication.

Attribute	Default	Description
statistics-enabled	false	Indicates whether or not this protocol will collect statistics, overriding the stack configuration.
thread-factory		The thread factory to use for handling asynchronous transport-specific tasks. <i>Deprecated: Configure the predefined internal thread pool instead.</i>
timer-executor		The thread pool executor to handle protocol-related timing tasks. <i>Deprecated: Configure the predefined timer thread pool instead.</i>

thread-pool Attributes

Table A.120. thread-pool Attributes

Attribute	Default	Description
keepalive-time	5000L	The amount of milliseconds that pool threads should be kept running when idle. If not specified, then threads will run until the executor is shut down.
max-threads	4	The maximum thread pool size.
min-threads	2	The core thread pool size, which is smaller than max-threads . If undefined, the core thread pool size is the same as max-threads .
queue-length	500	The queue length.

A.34. JGROUPS PROTOCOLS

Protocol	Protocol Type	Description
ASYM_ENCRYPT	Encryption	Uses a secret key, stored in a coordinator on the cluster, for encrypting messages between cluster members.
AUTH	Authentication	Provides a layer of authentication to cluster members.
azure.AZURE_PIN G	Discovery	Supports node discovery using Microsoft Azure's blob storage.
FD_ALL	Failure Detection	Provides failure detection based on a simple heartbeat protocol.
FD SOCK	Failure Detection	Provides failure detection based on a ring of TCP sockets created between cluster members.

Protocol	Protocol Type	Description
JDBC_PING	Discovery	Discovers cluster members by using a shared database where members write their address.
MERGE3	Merge	Merges the subclusters together in the event of a cluster split.
MFC	Flow Control	Provides multicast flow control between a sender and all cluster members.
MPING	Discovery	Discovers cluster members with IP multicast.
pbcast.GMS	Group Membership	Handles group membership, including new members joining the cluster, leave requests by existing members, and SUSPECT messages for crashed members.
pbcast.NAKACK2	Message Transmission	Ensures message reliability and order, guaranteeing that all messages sent by one sender will be received in the order they were sent.
pbcast.STABLE	Message Stability	Deletes messages that have been seen by all members.
PING	Discovery	Initial discovery of members, with support for dynamic discovery of cluster members.
SASL	Authentication	Provides a layer of authentication to cluster members using SASL mechanisms.
SYM_ENCRYPT	Encryption	Uses a shared keystore for encrypting messages between cluster members.
S3_PING	Discovery	Uses Amazon S3 to discover initial members.
TCPGOSSIP	Discovery	Discovers cluster members by using an external gossip router.
TCPPING	Discovery	Contains a static list of cluster member's addresses to form the cluster.
UFC	Flow Control	Provides unicast flow control between a sender and all cluster members
UNICAST3	Message Transmission	Ensures message reliability and order for unicast messages, guaranteeing that all messages sent by one sender will be received in the order they were sent.
VERIFY_SUSPECT	Failure Detection	Verifies that a suspected member has died by pinging the member one final time before evicting it.

Generic Protocol Attributes

All of the protocols have access to the following attributes.

Table A.121. protocol Attributes

Attribute	Default	Description
module	org.jgroups	The module with which to resolve the protocol type.
properties		Properties of this protocol.
statistics-enabled	false	Whether statistics are enabled.

Authentication Protocols

The authentication protocols are used to perform authentication, and are primarily responsible for ensuring that only authenticated members can join the cluster. These protocols sit below the **GMS** protocol, so that they may listen for requests to join the cluster.

- [AUTH](#)
- [SASL](#)

AUTH Attributes

While the **AUTH** protocol contains no additional attributes, it must have a token defined as a child element.



NOTE

When defining this protocol, the **auth-protocol** element is used instead of the **protocol** element.

Token Types

When using Elytron for security, it is recommended to use one of the following authentication tokens. These authentication tokens were intentionally designed for use with Elytron, and may not be used with legacy security configurations.

Table A.122. Elytron Token Types

Token	Description
cipher-token	An authentication token where the shared secret is transformed. RSA is the default algorithm used for the transformation.
digest-token	An authentication token where the shared secret is transformed. SHA-256 is the default algorithm used for the transformation.
plain-token	An authentication token with no additional transformations to the shared secret.

The following authentication tokens are inherited from JGroups, and are eligible for use in any configuration where authentication is desired.

Table A.123. JGroups Token Types

Token	Description
MD5Token	An authentication token where the shared secret is encrypted using either an MD5 or SHA hash. MD5 is the default algorithm used for the encryption.
SimpleToken	An authentication token with no additional transformations to the shared secret. This token is case-insensitive, and case is not considered when determining if strings match.
X509Token	An authentication token where the shared secret is encrypted using an X509 certificate.

SASL Attributes

Table A.124. SASL Attributes

Attribute	Default	Description
client_callback_handler		The class name of the CallbackHandler to use when a node acts as a client.
client_name		The name to use when a node acts as a client. This name will also be used to obtain the subject if using a JAAS login module.
client_password		The password to use when a node acts as a client. This password will also be used to obtain the subject if using a JAAS login module.
login_module_name		The name of the JAAS login module to use as a subject for creating the SASL client and server. This attribute is only required by certain mech values, such as GSSAPI.
mech		The name of the SASL authentication mechanism. This name can be any mechanism supported by the local SASL provider, and the JDK supplies CRAM-MD5 , DIGEST-MD5 , GSSAPI , and NTLM by default.
sasl_props		Properties of the defined mech .
server_callback_handler		The class name of the CallbackHandler to use when a node acts as a server.
server_name		The fully qualified server name.

Attribute	Default	Description
timeout	5000	The number of milliseconds to wait for a response to a challenge.

Discovery Protocols

The following protocols are used to find an initial membership for the cluster, which can then be used to determine the current coordinator. A list of the discovery protocols are below.

- [AZURE_PING](#)
- [JDBC_PING](#)
- MPING
- PING
- [S3_PING](#)
- [TCPGOSSIP](#)
- [TCPPING](#)

AZURE_PING Attributes

Table A.125. AZURE_PING Attributes

Attribute	Default	Description
container		The name of the blob container to use for PING data. This must be a valid DNS name.
storage_access_key		The secret access key for the storage account.
storage_account_name		The name of the Microsoft Azure storage account that contains your blob container.

JDBC_PING Attributes

Table A.126. JDBC_PING Attributes

Attribute	Default	Description
data-source		Datasource reference, to be used instead of the connection and JNDI lookup properties.



NOTE

When defining a **JDBC_PING** protocol, the **jdbc-protocol** element is used instead of the **protocol** element.

S3_PING Attributes

Table A.127. S3_PING Attributes

Attribute	Default	Description
access_key		The Amazon S3 access key used to access an S3 bucket.
host	s3.amazonaws.com	Destination of the S3 web service.
location		Name of the Amazon S3 bucket to use. The bucket must exist and use a unique name.
pre_signed_delete_url		The pre-signed URL to be used for the DELETE operation.
port	<ul style="list-style-type: none"> 443 if use_ssl is true. 80 if use_ssl is false. 	The port on which the web service is listening.
pre_signed_put_url		The pre-signed URL to be used for the PUT operation.
prefix		If set, and location is set, define the bucket name as PREFIX-LOCATION . If set, and a bucket does not exist at the specified PREFIX-LOCATION , then the bucket name becomes PREFIX followed by a random UUID.
secret_access_key		The Amazon S3 secret access key used to access an S3 bucket.
use_ssl	true	Determines if SSL is used when contacting the host and port combination.

TCPGOSSIP Attributes

Table A.128. TCPGOSSIP Attributes

Attribute	Default	Description
socket-binding		The socket binding specification for this protocol layer. <i>Deprecated: Use socket-bindings instead.</i>

Attribute	Default	Description
socket-bindings		The outbound socket bindings for this protocol.

**NOTE**

When defining a **TCPGOSSIP** protocol, the **socket-discovery-protocol** element is used instead of the **protocol** element.

TCPPING Attributes

Table A.129. TCPPING Attributes

Attribute	Default	Description
socket-binding		The socket binding specification for this protocol layer. <i>Deprecated: Use socket-bindings instead.</i>
socket-bindings		The outbound socket bindings for this protocol.

**NOTE**

When defining a **TCPPING** protocol, the **socket-discovery-protocol** element is used instead of the **protocol** element.

Encrypt Protocols

The following protocols are used to secure the communication stack. Encryption is based on a shared secret key that all members of the cluster have. This key is either acquired from a shared keystore, when using **SYM_ENCRYPT** or from a public/private key exchange, when using **ASYM_ENCRYPT**. When defining any of the following protocols an **encrypt-protocol** element is created in the resulting XML.

**NOTE**

If using **ASYM_ENCRYPT**, then the same stack must have an **AUTH** protocol defined. The **AUTH** protocol is optional when using **SYM_ENCRYPT**.

- [ASYM_ENCRYPT](#)
- [SYM_ENCRYPT](#)

ASYM_ENCRYPT Attributes

Table A.130. ASYM_ENCRYPT Attributes

Attribute	Default	Description
key-alias		The alias of the encryption key from the specified keystore.

Attribute	Default	Description
key-credential-reference		The credentials required to obtain the encryption key from the keystore.
key-store		A reference to a keystore containing the encryption key.

SYM_ENCRYPT Attributes

Table A.131. SYM_ENCRYPT Attributes

Attribute	Default	Description
key-alias		The alias of the encryption key from the specified keystore.
key-credential-reference		The credentials required to obtain the encryption key from the keystore.
key-store		A reference to a keystore containing the encryption key.

Failure Detection Protocols

The following protocols are used to probe members of the cluster to determine if they are still alive. These protocols do not have any additional attributes beyond the generic attributes.

- FD_ALL
- FD_SOCKET
- VERIFY_SUSPECT

Flow Control Protocols

The following protocols are responsible for flow control, or the process of adjusting the rate of a message sender to the slowest receiver. If a sender continuously sends messages at a rate faster than the receiver, then the receivers will either queue up or discard messages, resulting in retransmissions. These protocols do not have any additional attributes beyond the generic attributes.

- MFC - Multicast Flow Control
- UFC - Unicast Flow Control

Group Membership Protocols

The **pbcast.GMS** protocol is responsible for new members joining the cluster, existing members leaving the cluster, and members that are suspected of having crashed. This protocol does not have any additional attributes beyond the generic attributes.

Merge Protocols

If the cluster becomes split, then the **MERGE3** protocol is responsible for merging the subclusters back

together. While this protocol is responsible for merging the cluster members back together, this will not merge the state of the cluster. The application is responsible for handling the callback to merge states. This protocol does not have any additional attributes beyond the generic attributes.

Message Stability

The **pbcast.STABLE** protocol is responsible for garbage collecting messages that have been seen by all members of the cluster. This protocol initiates a stable message containing message numbers for a given member, called a digest. Once all members of the cluster have received the others' digests, then the message may be removed from the retransmission table. This protocol does not have any additional attributes beyond the generic attributes.

Reliable Message Transmission

The following protocols provide reliable message delivery and FIFO properties for messages sent to all nodes in a cluster. Reliable delivery means that no messages sent by a sender will ever be lost, as all messages are numbered, and retransmission requests are sent if a sequence number is not received. These protocols do not have any additional attributes beyond the generic attributes.

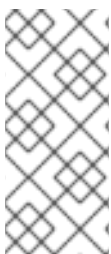
- `pbcast.NAKACK2`
- `pbcast.UNICAST3`

Deprecated Protocols

The following protocols have been deprecated, and have been replaced by a protocol that contains only the class name. For instance, instead of specifying **org.jgroups.protocols.ASYM_ENCRYPT**, the protocol name would be **ASYM_ENCRYPT**.

- `org.jgroups.protocols.ASYM_ENCRYPT`
- `org.jgroups.protocols.AUTH`
- `org.jgroups.protocols.JDBC_PING`
- `org.jgroups.protocols.SYM_ENCRYPT`
- `org.jgroups.protocols.TCPGOSSIP`
- `org.jgroups.protocols.TCPPING`

A.35. MICROPROFILE CONFIG SMALLRYE SUBSYSTEM ATTRIBUTES



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-microprofile-config-smallrye_1_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.132. MicroProfile Config SmallRye Subsystem:config-source Attributes

Attribute	Description
<code>class</code>	The ConfigSource class to load. The ConfigSource provides a source for the configuration values.

Attribute	Description
dir	The directory to scan. This directory contains properties files where each file name is the name of the property and the file content is its value.
ordinal	The ordinal value, or priority, for the ConfigSource . The values provided by ConfigSources with higher ordinal values override those provided by ConfigSources with lower ordinal values.
properties	Properties to store directly in the ConfigSource .

Table A.133. MicroProfile Config SmallRye Subsystem: **config-source-provider** Attributes

Attribute	Description
class	The ConfigSourceProvider class to load. The ConfigSourceProvider registers implementations for multiple ConfigSource instances.

A.36. APACHE HTTP SERVER MOD_CLUSTER DIRECTIVES

The mod_cluster connector is an Apache HTTP Server-based load balancer. It uses a communication channel to forward requests from the Apache HTTP Server to one of a set of application server nodes. The following directives can be set to configure mod_cluster.



NOTE

There is no need to use *ProxyPass* directives because mod_cluster automatically configures the URLs that must be forwarded to Apache HTTP Server.

Table A.134. mod_cluster Directives

Directive	Description	Values
CreateBalancers	Defines how the balancers are created in the Apache HTTP Server VirtualHosts. This allows directives like: ProxyPass /balancer://mycluster1/ .	<ul style="list-style-type: none"> ● 0: Create all VirtualHosts defined in Apache HTTP Server ● 1: Do not create balancers (at least one <i>ProxyPass</i> or <i>ProxyMatch</i> is required to define the balancer names) ● 2: Create only the main server (<i>default</i>)

Directive	Description	Values
UseAlias	Check that the alias corresponds to the server name.	<ul style="list-style-type: none"> ● 0: Ignore aliases (<i>default</i>) ● 1: Check aliases
LBstatusRecalTime	Time interval in seconds for load-balancing logic to recalculate the status of a node.	Default: 5 seconds
WaitBeforeRemove	Time in seconds before a removed node is forgotten by httpd.	Default: 10 seconds
ProxyPassMatch/ProxyPass	<p>ProxyPassMatch and ProxyPass are <i>mod_proxy</i> directives which, when using ! instead of the back-end URL, prevent reverse-proxy in the path. This is used to allow Apache HTTP Server to serve static content. For example:</p> <p>ProxyPassMatch ^(/.*\..gif)\$!</p> <p>This example allows the Apache HTTP Server to serve the .gif files directly.</p>	

**NOTE**

Due to performance optimizations for sessions in JBoss EAP 7, configuring hot-standby nodes is not supported.

mod_manager

The context of a `mod_manager` directive is `VirtualHost` in all cases, except when mentioned otherwise. *server config* context implies that the directive must be outside a `VirtualHost` configuration. If not, an error message is displayed and the Apache HTTP Server does not start.

Table A.135. mod_manager Directives

Directive	Description	Values
EnableMCPMReceive	<p>Allow the <code>VirtualHost</code> to receive the <i>MCPM</i> from the nodes.</p> <p>Include <i>EnableMCPMReceive</i> in the Apache HTTP Server configuration to allow <code>mod_cluster</code> to work. Save it in the <code>VirtualHost</code> where you configure advertising.</p>	

Directive	Description	Values
MemManagerFile	The base name for the names that <i>mod_manager</i> uses to store configuration, generate keys for shared memory or locked files. This must be an absolute path name; the directories are created if needed. It is recommended that these files are placed on a local drive and not an NFS share. Context: server config	\$server_root/logs/
Maxcontext	The maximum number of contexts supported by <i>mod_cluster</i> . Context: server config	Default: 100
Maxnode	The maximum number of nodes supported by <i>mod_cluster</i> . Context: server config	Default: 20
Maxhost	The maximum number of hosts, or aliases, supported by <i>mod_cluster</i> . It also includes the maximum number of balancers. Context: server config	Default: 20
Maxsessionid	The number of active <i>sessionid</i> stored to provide the number of active sessions in the <i>mod_cluster-manager</i> handler. A session is inactive when <i>mod_cluster</i> does not receive any information from the session within 5 minutes. Context: server config. This field is for demonstration and debugging purposes only.	0 : the logic is not activated.
MaxMCMPMaxMessSize	The maximum size of MCMP messages from other Max directives	Calculated from other Max directives. Min: 1024
ManagerBalancerName	The name of balancer to use when the JBoss EAP instance does not provide a balancer name.	<i>mycluster</i>

Directive	Description	Values
PersistSlots	Tells <code>mod_slotmem</code> to persist nodes, aliases and contexts in files. Context: server config	Off
CheckNonce	Switch check of <i>nonce</i> when using <i>mod_cluster-manager</i> handler.	on/off Default: on – Nonce checked
AllowDisplay	Switch additional display on <i>mod_cluster-manager</i> main page.	on/off Default: off – only version is displayed
AllowCmd	Allow commands using <i>mod_cluster-manager</i> URL.	on/off Default: on – Commands allowed
ReduceDisplay	Reduce the information displayed on the main <i>mod_cluster-manager</i> page, so that more nodes can be displayed on the page.	on/off Default: off – full information is displayed
SetHandler mod_cluster-manager	<p>Displays information about the node that <code>mod_cluster</code> sees from the cluster. The information includes generic information and additionally counts the number of active sessions.</p> <pre> <Location /mod_cluster- manager> SetHandler mod_cluster- manager Require ip 127.0.0.1 </Location> </pre>	on/off Default: off



NOTE

When accessing the location defined in **httpd.conf**:

- Transferred: Corresponds to the POST data sent to the back-end server.
- Connected: Corresponds to the number of requests that have been processed when the `mod_cluster` status page was requested.
- Num_sessions: Corresponds to the number of sessions `mod_cluster` report as active (on which there was a request within the past 5 minutes). This field is not present when `Maxsessionid` is zero and is for demonstration and debugging purposes only.

A.37. MODCLUSTER SUBSYSTEM ATTRIBUTES

The **modcluster** subsystem has the following structure:

- [proxy](#)
 - [load-provider=dynamic](#)
 - [custom-load-metric](#)
 - [load-metric](#)
 - [load-provider=simple](#)
 - [ssl](#)

The **load-provider=dynamic** resource allows you to configure factors, such as CPU, sessions, heap, memory, and weight to determine the load balancing behavior.

The **load-provider=simple** resource allows setting only a static constant as the factor attribute. This helps when the user does not need dynamic or complex rules to load balance the incoming HTTP requests.



NOTE

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/jboss-as-mod-cluster_3_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.136. proxy Configuration Options

Attribute	Default	Description
advertise	true	Whether to enable multicast-based advertise mechanism.
advertise-security-key		It is a shared secret between an httpd instance and the JBoss EAP servers listening for advertisements from the httpd instance.
advertise-socket		The name of the balancer on the reverse proxy to register with.
auto-enable-contexts	true	If set to false , contexts are registered with the reverse proxy as disabled. You can enable the context using the enable-context operation or by using the <code>mod_cluster_manager</code> console.
balancer		The name of the balancer on the reverse proxy to register with. If not set, the value is configured on the Apache HTTP Server side with the ManagerBalancerName directive, which defaults to mycluster .
connector		The name of Undertow listener that <code>mod_cluster</code> reverse proxy will connect to.

Attribute	Default	Description
excluded-contexts		A list of contexts to exclude from registration with the reverse proxies. If no host is indicated, the host is assumed to be localhost . ROOT indicates the root context of the web application.
flush-packets	false	Whether or not to enable packet flushing to the web server.
flush-wait	-1	Time to wait before flushing packets in httpd. Max value is 2,147,483,647 .
listener		The name of the Undertow listener that will be registered with the reverse proxy.
load-balancing-group		If set, requests are sent to the specified load balancing group on the load balancer.
max-attempts	1	The number of times the reverse proxy will attempt to send a given request to a worker before giving up.
node-timeout	-1	Timeout, in seconds, for proxy connections to a worker. This is the time that mod_cluster will wait for the back-end response before returning an error. If the node-timeout attribute is undefined, the httpd ProxyTimeout directive is used. If ProxyTimeout is undefined, the httpd Timeout directive is used, which defaults to 300 seconds.
ping	10	Time, in seconds, in which to wait for a pong answer to a ping.
proxies		List of proxies for mod_cluster to register with defined by outbound-socket-binding in socket-binding-group .
proxy-list		List of proxies. The format is HOST_NAME:PORT , separated with commas. <i>Deprecated in favor of proxies.</i>
proxy-url	/	Base URL for MCMP requests.

Attribute	Default	Description
session-draining-strategy	DEFAULT	<p>Session draining strategy used during undeployment of a web application. Valid values are DEFAULT, ALWAYS, or NEVER.</p> <p>DEFAULT Drain sessions before web application undeploy only if the web application is non-distributable.</p> <p>ALWAYS Always drain sessions before web application undeploy, even for distributable web applications.</p> <p>NEVER Do not drain sessions before web application undeploy.</p>
load-provider=simple		A load provider to use if no dynamic load provider is present. It assigns each cluster member a load factor of 1 , and distributes work evenly without applying a load balancing algorithm.
smax	-1	Soft maximum idle connection count in httpd.
socket-timeout	20	Number of seconds to wait for a response from an httpd proxy to MCMP commands before timing out, and flagging the proxy as in error.
ssl-context		Reference to the SSLContext to be used by mod_cluster.
status-interval	10	Number of seconds a STATUS message is sent from the application server to the reverse proxy. Allowed values are between 1 and 2,147,483,647 .
sticky-session	true	Whether subsequent requests for a given session should be routed to the same node, if possible.
sticky-session-force	false	Whether the reverse proxy should return an error in the event that the balancer is unable to route a request to the node to which it is stuck. This setting is ignored if sticky sessions are disabled.
sticky-session-remove	false	Remove session information on failover.
stop-context-timeout	10	The maximum time, in seconds, to wait for a context to process pending requests, for a distributable context, or to destroy active sessions, for a non-distributable context.

Attribute	Default	Description
ttl	-1	Time to live, in seconds, for idle connections above smax. Allowed values are between -1 and 2,147,483,647 .
worker-timeout	-1	Timeout to wait in httpd for an available worker to process the requests. Allowed values are between -1 and 2,147,483,647 .

Table A.137. load-provider=dynamic Configuration Options

Attribute	Default	Description
decay	2	The decay.
history	9	The history.
initial-load	0	<p>The initial load reported by a node. The valid range is 0-100, with 0 indicating maximum load.</p> <p>This attribute helps to gradually increase the load value of a newly joined node to avoid overloading it while joining a cluster.</p> <p>You can disable this behavior by setting the value as -1. When disabled, the node will report a load value of 100, indicating that it has no load when joining a cluster.</p>

Table A.138. custom-load-metric Attribute Options

Attribute	Default	Description
capacity	1.0	The capacity of the metric.
class		The class name of the custom metric.
property		The properties for the metric.
weight	1	The weight of the metric.

Table A.139. load-metric Attribute Options

Attribute	Default	Description
capacity	1.0	The capacity of the metric.

Attribute	Default	Description
property		The properties for the metric.
type		The type of the metric. Valid values are cpu , mem , heap , sessions , receive-traffic , send-traffic , requests , or busyness .
weight	1	The weight of the metric.

Table A.140. ssl Attribute Options

Attribute	Default	Description
ca-certificate-file		Certificate authority.
ca-revocation-url		Certificate authority revocation list.
certificate-key-file	\${user.home}/.keystore	Key file for the certificate.
cipher-suite		The allowed cipher suite.
key-alias		The key alias.
password	changeit	Password.
protocol	TLS	The SSL protocols that are enabled.

A.38. MOD_JK WORKER PROPERTIES

The **workers.properties** file defines the behavior of the workers to which mod_jk passes client requests. The **workers.properties** file defines where the different application servers are located and the way the workload should be balanced across them.

The general structure of a property is **worker.WORKER_NAME.DIRECTIVE**. The **WORKER_NAME** is a unique name that must match the **instance-id** configured in the JBoss EAP [undertow](#) subsystem. The **DIRECTIVE** is the setting to be applied to the worker.

Configuration Reference for Apache mod_jk Load Balancers

Templates specify default per-load-balancer settings. You can override the template within the load-balancer settings itself.

Table A.141. Global properties

Property	Description
worker.list	A comma separated list of worker names that will be used by <i>mod_jk</i> .

Table A.142. Mandatory Directives

Property	Description
type	The type of worker. The default type is ajp13 . Other possible values are ajp14 , lb , status . For more information on these directives, see the Apache Tomcat Connectors Reference at https://tomcat.apache.org/connectors-doc/reference/workers.html .

Table A.143. Load Balancing Directives

Property	Description
balance_workers	Specifies the worker nodes that the load balancer must manage. You can use the directive multiple times for the same load balancer. It consists of a comma-separated list of worker node names.
sticky_session	Specifies whether requests from the same session are always routed to the same worker. The default is 1 , meaning that sticky sessions are enabled. To disable sticky sessions, set it to 0 . Sticky sessions should usually be enabled, unless all of your requests are truly stateless.

Table A.144. Connection Directives

Property	Description
host	The host name or IP address of the back-end server. The back-end server must support the ajp protocol stack. The default value is localhost .
port	The port number of the back-end server instance listening for defined protocol requests. The default value is 8009 , which is the default listening port for AJP13 workers. The default value for AJP14 workers is 8011 .

Property	Description
ping_mode	<p>The conditions under which connections are probed for network status. The probe uses an empty AJP13 packet for CPing, and expects a CPong in response. Specify the conditions by using a combination of directive flags. The flags are not separated by a comma or any white-space. The ping_mode can be any combination of C, P, I, and A.</p> <ul style="list-style-type: none"> ● C - Connect. Probe the connection one time after connecting to the server. Specify the timeout using the value of connect_timeout. Otherwise, the value of ping_timeout is used. ● P - Prepost. Probe the connection before sending each request to the server. Specify the timeout using the prepost_timeout directive. Otherwise, the value of ping_timeout is used. ● I - Interval. Probe the connection at an interval specified by connection_ping_interval, if present. Otherwise, the value of ping_timeout is used. ● A - All. A shortcut for CPI, which specifies that all connection probes are used.
ping_timeout, connect_timeout, prepost_timeout, connection_ping_interval	<p>The timeout values for the connection probe settings above. The value is specified in milliseconds, and the default value for ping_timeout is 10000.</p>
lbfactor	<p>Specifies the load-balancing factor for an individual back-end server instance. This is useful to give a more powerful server more of the workload. To give a worker 3 times the default load, set this to 3: worker.my_worker.lbfactor=3</p>

The example below demonstrates load balancing with sticky sessions between two worker nodes, **node1** and **node2**, listening on port **8009**.

Example: workers.properties File

```
# Define list of workers that will be used for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host= node2.mydomain.com
worker.node2.type=ajp13
```



```

worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status

```

Further configuration details for Apache mod_jk are out of the scope of this document and can be found in the [Apache documentation](#).

A.39. SECURITY MANAGER SUBSYSTEM ATTRIBUTES

The **security-manager** subsystem itself does not have configurable attributes, but it has one child resource with configurable attributes: **deployment-permissions=default**.



NOTE

Attribute names in this table are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at ***EAP_HOME/docs/schema/wildfly-security-manager_1_0.xsd*** to view the elements as they appear in the XML, as there may be differences from the management model.

Table A.145. deployment-permissions Configuration Options

Attribute	Description
maximum-permissions	The maximum set of permissions that can be granted to a deployment or JARs.
minimum-permissions	The minimum set of permissions to be granted to a deployment or JARs.

A.40. INSTALL OPENSLL FROM JBOSS CORE SERVICES

The JBoss Core Services OpenSSL files can be installed either from the [ZIP](#) or from the [RPM](#) distributions. Follow the below steps depending on your installation method of choice.



NOTE

On Red Hat Enterprise Linux 8, standard system OpenSSL is supported thus installation of OpenSSL from JBoss Core Services is not necessary anymore.

Using JBoss Core Services OpenSSL ZIP File Distribution

**NOTE**

The path to **libs/** directory in the ZIP archive is **jbcS-openssl-*VERSION*/openssl/lib(64)** for Linux and **jbcS-openssl-*VERSION*/openssl/bin** for Windows.

1. Download the OpenSSL package from the [Software Downloads](#) page that pertains to your operating system and architecture.
2. Extract the downloaded ZIP file to your installation directory.
3. Notify JBoss EAP where to find the OpenSSL libraries.
You can do this using either of the following methods. In each of the following commands, be sure to replace **JBCS_OPENSSL_PATH** with the path to the JBoss Core Services OpenSSL libraries, for example, **/opt/rh/jbcS-httpd24/root/usr/lib64**.

- You can add the OpenSSL path to the **JAVA_OPTS** variable in the **standalone.conf** or **domain.conf** configuration file using the following argument.

```
JAVA_OPTS="$JAVA_OPTS -Dorg.wildfly.openssl.path=JBCS_OPENSSL_PATH
```

- You can define a system property that specifies the OpenSSL path using the following management CLI command.

```
/system-property=org.wildfly.openssl.path:add(value=JBCS_OPENSSL_PATH)
```

**IMPORTANT**

Regardless of the method you use, you must perform a server restart for either the **JAVA_OPTS** value or the system property to take effect. A server reload is not sufficient.

Using JBoss Core Services OpenSSL RPM Distribution

1. Ensure that the system is registered to the JBoss Core Services channel:
 - a. Determine the JBoss Core Services CDN repository name for your operating system version and architecture:
 - **RHEL 6:** `jb-coreservices-1-for-rhel-6-server-rpms`
 - **RHEL 7:** `jb-coreservices-1-for-rhel-7-server-rpms`

- b. Enable the repository on the system:

```
# subscription-manager repos --enable REPO_NAME
```

- c. Ensure the following message is seen:

```
Repository REPO_NAME is enabled for this system.
```

2. Install OpenSSL from this channel:

```
# yum install jbcS-httpd24-openssl
```

3. Once the installation completes, the JBoss OpenSSL libraries will be available in **/opt/rh/jbcs-httpd24/root/usr/lib64**, or just **/opt/rh/jbcs-httpd24/root/usr/lib** on x86 architecture.
4. Notify JBoss EAP where to find the OpenSSL libraries.
You can do this using either of the following methods. In each of the following commands, be sure to replace **JBCS_OPENSSL_PATH** with the path to the JBoss Core Services OpenSSL libraries, for example, **/opt/rh/jbcs-httpd24/root/usr/lib64**.

- You can update the **WILDFLY_OPTS** variable for the **eap7-standalone** or **eap7-domain** settings in the service configuration file.

```
WILDFLY_OPTS="$WILDFLY_OPTS -
Dorg.wildfly.openssl.path=JBCS_OPENSSL_PATH"
```

- You can define a system property that specifies the OpenSSL path using the following management CLI command.

```
/system-property=org.wildfly.openssl.path:add(value=JBCS_OPENSSL_PATH)
```



IMPORTANT

Regardless of the method you use, you must perform a server restart for either the **WILDFLY_OPTS** value or the system property to take effect. A server reload is not sufficient.

A.41. CONFIGURE JBOSS EAP TO USE OPENSSL

There are multiple ways in which you can configure JBoss EAP to use OpenSSL:

- You can reconfigure the **elytron** subsystem to give OpenSSL priority so that it is used in all cases by default.



NOTE

Although OpenSSL is installed in the **elytron** subsystem, it is not the default TLS provider.

```
/subsystem=elytron:write-attribute(name=initial-providers, value=combined-providers)
/subsystem=elytron:undefine-attribute(name=final-providers)

reload
```

- In the **elytron** subsystem, the OpenSSL provider can also be specified on the **ssl-context** resource. That way, the OpenSSL protocol can be selected on a case-by-case basis instead of using the default priority.
To create the **ssl-context** resource and use the OpenSSL libraries in your Elytron-based SSL/TLS configuration, use the following command.

```
/subsystem=elytron/server-ssl-context=httpsSSC:add(key-manager=localhost-manager,
trust-manager=ca-manager, provider-name=openssl)

reload
```

- To use the OpenSSL libraries in your legacy **security** subsystem SSL/TLS configuration:

```
/core-service=management/security-realm=ApplicationRealm/server-identity=ssl:write-  
attribute(name=protocol,value=openssl.TLSv1.2)  
  
reload
```

The different OpenSSL protocols that can be used are:

- openssl.TLS
- openssl.TLSv1
- openssl.TLSv1.1
- openssl.TLSv1.2

JBoss EAP will automatically try to search for the OpenSSL libraries on the system and use them. You can also specify a custom OpenSSL libraries location by using the **org.wildfly.openssl.path** property during JBoss EAP startup. Only the OpenSSL library version 1.0.2 or greater provided by JBoss Core Services is supported.

If OpenSSL is loaded properly, you will see a message in the **server.log** during JBoss EAP startup, similar to:

```
15:37:59,814 INFO [org.wildfly.openssl.SSL] (MSC service thread 1-7) WFOPENSSL0002 OpenSSL  
Version OpenSSL 1.0.2k-fips 23 Mar 2017
```

A.42. PLATFORM MODULES PROVIDED FOR JAVA 8

- **java.base**: This dependency is always included in the provided module loaders
- **java.compiler**
- **java.datatransfer**
- **java.desktop**
- **java.instrument**
- **java.jnlp**
- **java.logging**
- **java.management**
- **java.management.rmi**
- **java.naming**
- **java.prefs**
- **java.rmi**
- **java.scripting**

- **java.se**: This module alias aggregates the following set of base modules:
 - **java.compiler**
 - **java.datatransfer**
 - **java.desktop**
 - **java.instrument**
 - **java.logging**
 - **java.management**
 - **java.management.rmi**
 - **java.naming**
 - **java.prefs**
 - **java.rmi**
 - **java.scripting**
 - **java.security.jgss**
 - **java.security.sasl**
 - **java.sql**
 - **java.sql.rowset**
 - **java.xml**
 - **java.xml.crypto**
- **java.security.jgss**
- **java.security.sasl**
- **java.smartcardio**
- **java.sql**
- **java.sql.rowset**
- **java.xml**
- **java.xml.crypto**
- **javafx.base**
- **javafx.controls**
- **javafx.fxml**
- **javafx.graphics**

- `javafx.media`
- `javafx.swing`
- `javafx.web`
- `jdk.accessibility`
- `jdk.attach`
- `jdk.compiler`
- `jdk.httpserver`
- `jdk.jartool`
- `jdk.javadoc`
- `jdk.jconsole`
- `jdk.jdi`
- `jdk.jfr`
- `jdk.jsobject`
- `jdk.management`
- `jdk.management.cmm`
- `jdk.management.jfr`
- `jdk.management.resource`
- `jdk.net`
- `jdk.plugin.dom`
- `jdk.scripting.nashorn`
- `jdk.sctp`
- `jdk.security.auth`
- `jdk.security.jgss`
- `jdk.unsupported`
- `jdk.xml.dom`

Revised on 2021-03-18 15:43:58 UTC

