



JBoss Enterprise Application Platform 6 Administration and Configuration Guide

For Use with JBoss Enterprise Application Platform 6
Edition 3

Nidhi Chaudhary
Vikram Goyal
Scott Mumford
Keerat Verma

Russell Dickenson
Eamon Logue
David Ryan
Tom Wells

Sande Gilda
Darrin Mison
Misty Stanley-Jones

JBoss Enterprise Application Platform 6 Administration and Configuration Guide

For Use with JBoss Enterprise Application Platform 6 Edition 3

Nidhi Chaudhary

Russell Dickenson

Sande Gilda

Vikram Goyal

Eamon Logue

Darrin Mison

Scott Mumford

David Ryan

Misty Stanley-Jones

Keerat Verma

Tom Wells

Legal Notice

Copyright © 2013 Red Hat, Inc..

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This book is a guide to the administration and configuration of JBoss Enterprise Application Platform 6 and its patch releases.

Table of Contents

Preface	11
1. Document Conventions	11
1.1. Typographic Conventions	11
1.2. Pull-quote Conventions	12
1.3. Notes and Warnings	12
2. Getting Help and Giving Feedback	13
2.1. Do You Need Help?	13
2.2. Give us Feedback	13
Chapter 1. Introduction to Administering the JBoss Enterprise Application Platform ..	14
1.1. Introducing JBoss Enterprise Application Platform 6	14
1.2. New and Changed Features in JBoss Enterprise Application Platform 6	14
Chapter 2. Application Server Management ..	15
2.1. Manage the Application Server	15
2.2. Installation Structure and Details	15
2.3. JBoss Enterprise Application Platform 6 Profiles	16
2.4. About JBoss Enterprise Application Platform 6 Configuration Files	17
2.5. Management APIs	17
2.5.1. Management Application Programming Interfaces (APIs)	18
2.6. Start JBoss Enterprise Application Platform 6	19
2.6.1. Start JBoss Enterprise Application Platform 6	19
2.6.2. Start JBoss Enterprise Application Platform 6 as a Standalone Server	19
2.6.3. Start JBoss Enterprise Application Platform 6 as a Managed Domain	19
2.6.4. Start JBoss Enterprise Application Platform 6 with an Alternative Configuration	20
2.6.5. Stop JBoss Enterprise Application Platform 6	21
2.6.6. Reference of Switches and Arguments to pass at Server Runtime	21
2.7. Run JBoss Enterprise Application Platform 6 as a Service	22
2.7.1. Run JBoss Enterprise Application Platform 6 as an Operating System Service	22
2.7.2. Install JBoss Enterprise Application Platform as a Service in Red Hat Enterprise Linux	22
2.7.3. Install JBoss Enterprise Application Platform 6 as a Service in Microsoft Windows	23
2.8. Start and Stop Servers	24
2.8.1. Start and Stop Servers	24
2.8.2. Start a Server Using the Management Console	24
2.8.3. Stop a Server Using the Management Console	26
2.9. Filesystem Paths	28
2.9.1. Filesystem Paths	28
2.10. Configuration File History	29
2.10.1. Configuration File History	29
2.10.2. Start the Server with a Previous Configuration	29
2.10.3. Save a Configuration Snapshot Using the Management CLI	30
2.10.4. Load a Configuration Snapshot	30
2.10.5. Delete a Configuration Snapshot Using Management CLI	31
2.10.6. List All Configuration Snapshots Using Management CLI	31
Chapter 3. Management Interfaces ..	33
3.1. About the Management Console and Management CLI	33
3.2. The Management Console	33
3.2.1. Management Console	33
3.2.2. Log in to the Management Console	33
3.2.3. Change the Language of the Management Console	34
3.2.4. Configure a Server Using the Management Console	34

3.2.5. Add a Deployment in the Management Console	35
3.2.6. Create a New Server in the Management Console	38
3.2.7. Change the Default Log Levels Using the Management Console	38
3.2.8. Create a New Server Group in the Management Console	39
3.3. The Management CLI	41
3.3.1. About the Management Command Line Interface (CLI)	41
3.3.2. Launch the Management CLI	41
3.3.3. Quit the Management CLI	41
3.3.4. Connect to a Managed Server Instance Using the Management CLI	41
3.3.5. Get Help with the Management CLI	42
3.3.6. Use the Management CLI in Batch Mode	42
3.3.7. Use Operations and Commands in the Management CLI	43
3.3.8. Reference of Management CLI Commands	45
3.3.9. Reference of Management CLI Operations	46
3.4. Management CLI Operations	49
3.4.1. Display the Attributes of a Resource with the Management CLI	49
3.4.2. Display the Active User in the Management CLI	51
3.4.3. Display System and Server Information in the Management CLI	51
3.4.4. Display an Operation Description using the Management CLI	51
3.4.5. Display the Operation Names using the Management CLI	52
3.4.6. Display Available Resources using the Management CLI	53
3.4.7. Display Available Resource Descriptions using the Management CLI	59
3.4.8. Reload the Application Server using the Management CLI	60
3.4.9. Shut the Application Server down using the Management CLI	60
3.4.10. Configure an Attribute with the Management CLI	60
3.5. The Management CLI Command History	61
3.5.1. Use the Management CLI Command History	61
3.5.2. Show the Management CLI Command history	62
3.5.3. Clear the Management CLI Command history	62
3.5.4. Disable the Management CLI Command history	62
3.5.5. Enable the Management CLI Command history	63
Chapter 4. User Management	64
4.1. User Creation	64
4.1.1. Add the Initial User for the Management Interfaces	64
4.1.2. Add a User to the Management Interface	65
Chapter 5. Network and Port Configuration	66
5.1. Interfaces	66
5.1.1. About Interfaces	66
5.1.2. Configure Interfaces	67
5.2. Socket Binding Groups	69
5.2.1. About Socket Binding Groups	69
5.2.2. Configure Socket Bindings	73
5.2.3. Network Ports Used By JBoss Enterprise Application Platform 6	74
5.2.4. About Port Offsets for Socket Binding Groups	77
5.2.5. Configure Port Offsets	77
5.3. IPv6	78
5.3.1. Configure JVM Stack Preferences for IPv6 Networking	78
5.3.2. Configure the Interface Declarations for IPv6 Networking	78
5.3.3. Configure JVM Stack Preferences for IPv6 Addresses	79
Chapter 6. Datasource Management	80
6.1. Introduction	80
6.1.1. About JDBC	80
6.1.2. JBoss Enterprise Application Platform 6 Supported Databases	80

6.1.3. Types of Datasources	80
6.1.4. The Example Datasource	80
6.1.5. Deployment of -ds.xml files	80
6.2. JDBC Drivers	81
6.2.1. Install a JDBC Driver with the Management Console	81
6.2.2. Install a JDBC Driver as a Core Module	81
6.2.3. JDBC Driver Download Locations	82
6.2.4. Access Vendor Specific Classes	82
6.3. Non-XA Datasources	84
6.3.1. Create a Non-XA Datasource with the Management Interfaces	84
6.3.2. Modify a Non-XA Datasource with the Management Interfaces	85
6.3.3. Remove a Non-XA Datasource with the Management Interfaces	86
6.4. XA Datasources	87
6.4.1. Create an XA Datasource with the Management Interfaces	88
6.4.2. Modify an XA Datasource with the Management Interfaces	89
6.4.3. Remove an XA Datasource with the Management Interfaces	91
6.4.4. XA Recovery	92
6.4.4.1. About XA Recovery Modules	92
6.4.4.2. Configure XA Recovery Modules	92
6.5. Datasource Security	93
6.5.1. About Datasource Security	93
6.6. Datasource Configuration	94
6.6.1. Datasource Parameters	94
6.6.2. Datasource Connection URLs	99
6.6.3. Datasource Extensions	99
6.7. Example Datasources	100
6.7.1. Example PostgreSQL Datasource	100
6.7.2. Example PostgreSQL XA Datasource	101
6.7.3. Example MySQL Datasource	101
6.7.4. Example MySQL XA Datasource	102
6.7.5. Example Oracle Datasource	103
6.7.6. Example Oracle XA Datasource	104
6.7.7. Example Microsoft SQLServer Datasource	105
6.7.8. Example Microsoft SQLServer XA Datasource	106
6.7.9. Example IBM DB2 Datasource	107
6.7.10. Example IBM DB2 XA Datasource	108
6.7.11. Example Sybase Datasource	109
6.7.12. Example Sybase XA Datasource	110
Chapter 7. Configuring Modules	112
7.1. Introduction	112
7.1.1. Modules	112
7.1.2. Global Modules	112
7.1.3. Module Dependencies	112
7.1.4. Subdeployment Class Loader Isolation	113
7.2. Disable Sub-Deployment Module Isolation for All Deployments	113
7.3. Add a module to all deployments	114
7.4. Reference	115
7.4.1. Included Modules	115
7.4.2. Dynamic Module Naming	120
Chapter 8. Application Deployment	121
8.1. About Application Deployment	121
8.2. Deploy with the Management Console	121
8.2.1. Manage Application Deployment in the Management Console	121
8.2.2. Deploy an Application Using the Management Console	121

8.2.3. Undeploy an Application Using the Management Console	124
8.3. Deploy with the Management CLI	127
8.3.1. Manage Application Deployment in the Management CLI	127
8.3.2. Deploy an Application in a Managed Domain Using the Management CLI	128
8.3.3. Undeploy an Application in a Managed Domain Using the Management CLI	128
8.3.4. Deploy an Application in a Standalone Server Using the Management CLI	128
8.3.5. Undeploy an Application in a Standalone Server Using the Management CLI	129
8.4. Deploy with the Deployment Scanner	129
8.4.1. Manage Application Deployment in the Deployment Scanner	129
8.4.2. Deploy an Application to a Standalone Server Instance with the Deployment Scanner	129
8.4.3. Undeploy an Application to a Standalone Server Instance with the Deployment Scanner	
8.4.4. Redeploy an Application to a Standalone Server Instance with the Deployment Scanner	130
8.4.5. Reference for Deployment Scanner Marker Files	131
8.4.6. Reference for Deployment Scanner Attributes	132
8.4.7. Configure the Deployment Scanner	133
8.4.8. Configure the Deployment Scanner with the Management CLI	133
8.5. Deploy with Maven	136
8.5.1. Manage Application Deployment with Maven	136
8.5.2. Deploy an Application with Maven	136
8.5.3. Undeploy an Application with Maven	137
Chapter 9. Securing JBoss Enterprise Application Platform	139
9.1. About the Security Subsystem	139
9.2. About the Structure of the Security Subsystem	139
9.3. Configure the Security Subsystem	141
9.4. About Deep Copy Subject Mode	141
9.5. Enable Deep Copy Subject Mode	142
9.6. Security Domains	142
9.6.1. About Security Domains	142
9.6.2. About Picketbox	143
9.6.3. About Authentication	143
9.6.4. Configure Authentication in a Security Domain	143
9.6.5. About Authorization	144
9.6.6. Configure Authorization in a Security Domain	145
9.6.7. About Security Auditing	145
9.6.8. Configure Security Auditing	146
9.6.9. About Security Mapping	146
9.6.10. Configure Security Mapping in a Security Domain	146
9.6.11. Use a Security Domain in Your Application	147
9.6.12. Java Authorization Contract for Containers (JACC)	149
9.6.12.1. About Java Authorization Contract for Containers (JACC)	149
9.6.12.2. Configure Java Authorization Contract for Containers (JACC) Security	149
9.6.13. Java Authentication SPI for Containers (JASPI)	151
9.6.13.1. About Java Authentication SPI for Containers (JASPI) Security	151
9.6.13.2. Configure Java Authentication SPI for Containers (JASPI) Security	151
9.7. Management Interface Security	151
9.7.1. Default User Security Configuration	151
9.7.2. Overview of Advanced Management Interface Configuration	152
9.7.3. About LDAP	153
9.7.4. Use LDAP to Authenticate to the Management Interfaces	153
9.7.5. Disable the HTTP Management Interface	155
9.7.6. Remove Silent Authentication from the Default Security Realm	156
9.7.7. Disable Remote Access to the JMX Subsystem	157
9.7.8. Configure Security Realms for the Management Interfaces	158
9.8. Network Security	159

9.8.1. Secure the Management Interfaces	159
9.8.2. Specify Which Network Interface the JBoss Enterprise Application Platform Uses	159
9.8.3. Configure Network Firewalls to Work with JBoss Enterprise Application Platform	160
9.8.4. Network Ports Used By JBoss Enterprise Application Platform	162
9.9. Java Security Manager	164
9.9.1. About the Java Security Manager	164
9.9.2. Run JBoss Enterprise Application Platform Within the Java Security Manager	164
9.9.3. About Java Security Manager Policies	165
9.9.4. Write a Java Security Manager Policy	165
9.9.5. Debug Security Manager Policies	168
9.10. Application Security	168
9.10.1. Enabling/Disabling Descriptor Based Property Replacement	168
9.11. Password Vaults for Sensitive Strings	169
9.11.1. About Securing Sensitive Strings in Clear-Text Files	169
9.11.2. Create a Java Keystore to Store Sensitive Strings	169
9.11.3. Mask the Keystore Password and Initialize the Password Vault	171
9.11.4. Configure the JBoss Enterprise Application Platform to Use the Password Vault	172
9.11.5. Store and Retrieve Encrypted Sensitive Strings in the Java Keystore	173
9.11.6. Store and Resolve Sensitive Strings In Your Applications	175
Chapter 10. Security Administration Reference	178
10.1. Included Authentication Modules	178
10.2. Included Authorization Modules	201
10.3. Included Security Mapping Modules	201
10.4. Included Security Auditing Provider Modules	201
Chapter 11. The Logging Subsystem	203
11.1. Introduction	203
11.1.1. Overview of Logging	203
11.1.2. Application Logging Frameworks Supported By JBoss LogManager	203
11.1.3. Configure Boot Logging	203
11.1.4. Default Log File Locations	203
11.1.5. About Log Levels	204
11.1.6. Supported Log Levels	204
11.1.7. About Log Categories	205
11.1.8. About the Root Logger	205
11.1.9. About Log Handlers	205
11.1.10. Types of Log Handlers	205
11.1.11. About Log Formatters	206
11.1.12. Log Formatter Syntax	206
11.2. Configure Logging in the Management Console	206
11.3. Logging Configuration in the CLI	207
11.3.1. Configure the Root Logger with the CLI	207
11.3.2. Configure a Log Category in the CLI	209
11.3.3. Configure a Console Log Handler in the CLI	211
11.3.4. Configure a File Log Handler in the CLI	213
11.3.5. Configure a Periodic Log Handler in the CLI	216
11.3.6. Configure a Size Log Handler in the CLI	220
11.3.7. Configure a Async Log Handler in the CLI	223
11.4. Logging Configuration Properties	226
11.4.1. Root Logger Properties	226
11.4.2. Log Category Properties	226
11.4.3. Console Log Handler Properties	226
11.4.4. File Log Handler Properties	227
11.4.5. Periodic Log Handler Properties	227
11.4.6. Size Log Handler Properties	228

11.4.7. Async Log Handler Properties	229
11.5. Sample XML Configuration for Logging	229
11.5.1. Sample XML Configuration for the Root Logger	230
11.5.2. Sample XML Configuration for a Log Category	230
11.5.3. Sample XML Configuration for a Console Log Handler	230
11.5.4. Sample XML Configuration for a File Log Handler	230
11.5.5. Sample XML Configuration for a Periodic Log Handler	230
11.5.6. Sample XML Configuration for a Size Log Handler	231
11.5.7. Sample XML Configuration for a Async Log Handler	231
Chapter 12. JVM	232
12.1. About JVM	232
12.1.1. About JVM Settings	232
12.1.2. Display the JVM Status in the Management Console	233
Chapter 13. HTTP Clustering and Load Balancing	236
13.1. Introduction	236
13.1.1. About High-Availability and Load Balancing Clusters	236
13.1.2. Components Which Can Benefit from High Availability	236
13.1.3. Overview of HTTP Connectors	236
13.1.4. Worker Node	237
13.2. General Configuration	238
13.2.1. Subsystem Configuration Overview	238
13.2.2. Configure the Web Subsystem	238
13.2.3. Implement SSL Encryption for the JBoss Enterprise Application Platform Web Server	242
13.2.4. Generate a SSL Encryption Key and Certificate	243
13.2.5. SSL Connector Reference	246
13.2.6. About Web Service Endpoints	249
13.2.7. Replace the Default Welcome Web Application	249
13.2.8. About the Stand-Alone HTTPD	249
13.2.9. Install the Apache HTTPD included with JBoss Enterprise Application Platform 6	250
13.2.10. Use an External HTTPD as the Web Front-end for JBoss Enterprise Application Platform Applications	250
13.2.11. Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD	251
13.2.12. Use TCP Communication for the Clustering Subsystem	252
13.2.13. Configure the JGroups Subsystem to Use TCP	252
13.2.14. Configure the mod_cluster Subsystem to Use TCP	254
13.3. Web, HTTP Connectors, and HTTP Clustering	255
13.3.1. About the mod_cluster HTTP Connector	255
13.3.2. Configure the mod_cluster Subsystem	255
13.3.3. Install the mod_cluster Module Into Apache HTTPD or Enterprise Web Server HTTPD	255
13.3.4. Configure Server Advertisement Properties for Your mod_cluster-enabled HTTPD	264
13.3.5. Configure a mod_cluster Worker Node	267 266
13.4. Apache mod_jk	269
13.4.1. About the Apache mod_jk HTTP Connector	269
13.4.2. Configure the JBoss Enterprise Application Platform to Communicate with Apache Mod_jk	270
13.4.3. Install the Mod_jk Module Into Apache HTTPD or Enterprise Web Server HTTPD	270
13.4.4. Configuration Reference for Apache Mod_jk Workers	273 270
13.5. Apache mod_proxy	275
13.5.1. About the Apache mod_proxy HTTP Connector	275
13.5.2. Install the Mod_proxy HTTP Connector Into Apache HTTPD	275
13.6. Microsoft ISAPI	277
13.6.1. About the Internet Server API (ISAPI) HTTP Connector	277
13.6.2. Configure Microsoft IIS to Use the ISAPI Redirector	278
13.6.3. Configure the ISAPI Redirector to Send Client Requests to the JBoss Enterprise	

Application Platform	279
13.6.4. Configure ISAPI to Balance Client Requests Across Multiple JBoss Enterprise Application Platform Servers	281
13.7. Oracle NSAPI	283
13.7.1. About the Netscape Server API (NSAPI) HTTP Connector	283
13.7.2. Configure the NSAPI Connector on Oracle Solaris	284
13.7.3. Configure NSAPI as a Basic HTTP Connector	285
13.7.4. Configure NSAPI as a Load-balancing Cluster	286
Chapter 14. Messaging	289
14.1. HornetQ	289
14.1.1. HornetQ	289
14.1.2. About Java Messaging Service (JMS)	289
14.1.3. Supported Messaging Styles	289
14.1.4. About Acceptors and Connectors	289
14.1.5. About Bridges	290
14.1.6. Work with Large Messages	290
14.1.7. Configure High-availability (HA) Failover	291
14.1.8. Embed HornetQ in Applications	291
14.1.9. Configure the JMS Server	291
14.1.10. About Java Naming and Directory Interface (JNDI)	293
14.1.11. Configure JNDI for HornetQ	293
14.1.12. Configure JMS Address Settings	294
14.1.13. Reference for HornetQ Configuration Attributes	298
14.1.14. Configure Messaging with HornetQ	300
14.1.15. Configure Delayed Redelivery	300
14.1.16. Configure Dead Letter Addresses	301
14.1.17. Configure Message Expiry Addresses	301
14.1.18. Set Message Expiry	302
Chapter 15. Transaction Subsystem	303
15.1. Transaction Subsystem Configuration	303
15.1.1. Transactions Configuration Overview	303
15.1.2. Configure the Transaction Manager	303
15.1.3. Configure Your Datasource to Use JTA Transactions	307
15.1.4. Configure an XA Datasource	307
15.1.5. About Transaction Log Messages	308
15.1.6. Configure Logging for the Transaction Subsystem	308
15.2. Transaction Administration	309
15.2.1. Browse and Manage Transactions	309
15.3. Transaction References	313
15.3.1. JBoss Transactions Errors and Exceptions	313
15.3.2. JTA Clustering Limitations	313
15.4. ORB Configuration	313
15.4.1. About Common Object Request Broker Architecture (CORBA)	313
15.4.2. Configure the ORB for JTS Transactions	313
Chapter 16. Enterprise JavaBeans	315
16.1. Introduction	315
16.1.1. Overview of Enterprise JavaBeans	315
16.1.2. Overview of Enterprise JavaBeans for Administrators	315
16.1.3. Enterprise Beans	315
16.1.4. Session Beans	316
16.1.5. Message-Driven Beans	316
16.2. Configuring Bean Pools	316
16.2.1. Bean Pools	316

16.2.2. Create a Bean Pool	316
16.2.3. Remove a Bean Pool	318
16.2.4. Edit a Bean Pool	319
16.2.5. Assign Bean Pools for Session and Message-Driven Beans	320
16.3. Configuring EJB Thread Pools	321
16.3.1. Enterprise Bean Thread Pools	321
16.3.2. Create a Thread Pool	321
16.3.3. Remove a Thread Pool	323
16.3.4. Edit a Thread Pool	324
16.4. Configuring Session Beans	325
16.4.1. Session Bean Access Timeout	325
16.4.2. Set Default Session Bean Access Timeout Values	325
16.5. Configuring Message-Driven Beans	327
16.5.1. Set Default Resource Adapter for Message-Driven Beans	327
16.6. Configuring the EJB3 Timer Service	328
16.6.1. EJB3 Timer Service	328
16.6.2. Configure the EJB3 timer Service	328
16.7. Configuring the EJB Asynchronous Invocation Service	329
16.7.1. EJB3 Asynchronous Invocation Service	329
16.7.2. Configure the EJB3 Asynchronous Invocation Service Thread Pool	329
16.8. Configuring the EJB3 Remote Invocation Service	329
16.8.1. EJB3 Remote Service	330
16.8.2. Configure the EJB3 Remote Service	330
16.9. Configuring EJB 2.x Entity Beans	330
16.9.1. EJB Entity Beans	330
16.9.2. Container-Managed Persistence	330
16.9.3. Enable EJB 2.x Container-Managed Persistence	331
16.9.4. Configure EJB 2.x Container-Managed Persistence	331
16.9.5. CMP Subsystem Properties for HiLo Key Generators	332
Chapter 17. Java Connector Architecture (JCA)	333
17.1. Introduction	333
17.1.1. About the Java EE Connector API (JCA)	333
17.1.2. Java Connector Architecture (JCA)	333
17.1.3. Resource Adapters	333
17.2. Configure the Java Connector Architecture (JCA) Subsystem	333
17.3. Deploy a Resource Adapter	337
17.4. Configure a Deployed Resource Adapter	340
17.5. Resource Adapter Descriptor Reference	353
17.6. Deploy the WebSphere MQ Resource Adapter	357
Chapter 18. Deploy JBoss Enterprise Application Platform 6 on Amazon EC2	360
18.1. Introduction	360
18.1.1. About Amazon EC2	360
18.1.2. About Amazon Machine Instances (AMIs)	360
18.1.3. About JBoss Cloud Access	360
18.1.4. JBoss Cloud Access Features	360
18.1.5. Supported Amazon EC2 Instance Types	360
18.1.6. Supported Red Hat AMIs	361
18.2. Deploying JBoss Enterprise Application Platform 6 on Amazon EC2	361
18.2.1. Overview of Deploying JBoss Enterprise Application Platform 6 on Amazon EC2	361
18.2.2. Non-clustered JBoss Enterprise Application Platform 6	362
18.2.2.1. About Non-clustered Instances	362
18.2.2.2. Non-clustered Instances	362
18.2.2.2.1. Launch a Non-clustered JBoss Enterprise Application Platform 6 Instance	362
18.2.2.2.2. Deploy an Application on a non-clustered JBoss Enterprise Application Platform	

Instance	363
18.2.2.2.3. Test the Non-clustered JBoss Enterprise Application Platform 6 Instance	364
18.2.2.3. Non-clustered Managed Domains	364
18.2.2.3.1. Launch an Instance to Serve as a Domain Controller	365
18.2.2.3.2. Launch One or More Instances to Serve as Host Controllers	367
18.2.2.3.3. Test the Non-Clustered JBoss Enterprise Application Platform 6 Managed Domain	368
18.2.3. Clustered JBoss Enterprise Application Platform 6	369
18.2.3.1. About Clustered Instances	369
18.2.3.2. Create a Relational Database Service Database Instance	369
18.2.3.3. About Virtual Private Clouds	369
18.2.3.4. Create a Virtual Private Cloud (VPC)	370
18.2.3.5. Launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the VPC	370
18.2.3.6. Configure the VPC Private Subnet Default Route	372
18.2.3.7. About Identity and Access Management (IAM)	372
18.2.3.8. Configure IAM Setup	372
18.2.3.9. About the S3 Bucket	373
18.2.3.10. Configure S3 Bucket Setup	373
18.2.3.11. Clustered Instances	374
18.2.3.11.1. Launch Clustered JBoss Enterprise Application Platform 6 AMIs	374
18.2.3.11.2. Test the Clustered JBoss Enterprise Application Platform 6 Instance	377
18.2.3.12. Clustered Managed Domains	377
18.2.3.12.1. Launch an Instance to Serve as a Cluster Domain Controller	377
18.2.3.12.2. Launch One or More Instances to Serve as Cluster Host Controllers	380
18.2.3.12.3. Test the Clustered JBoss Enterprise Application Platform 6 Managed Domain	382
18.3. Establishing Monitoring with JBoss Operations Network (JON)	383
18.3.1. About AMI Monitoring	383
18.3.2. About Connectivity Requirements	383
18.3.3. About Network Address Translation (NAT)	383
18.3.4. About Amazon EC2 and DNS	384
18.3.5. About Routing in EC2	384
18.3.6. About Terminating and Restarting with JON	384
18.3.7. Configure an Instance to Register with JBoss Operations Network	385
18.4. User Script Configuration	385
18.4.1. Permanent Configuration Parameters	385
18.4.2. Custom Script Parameters	387
18.5. Troubleshooting	388
18.5.1. About Troubleshooting Amazon EC2	388
18.5.2. Diagnostic Information	388
Chapter 19. Supplemental References	389
19.1. Download Files From the Red Hat Customer Portal	389
19.2. Configure the Default JDK on Red Hat Enterprise Linux	389
Revision History	391

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the [Liberation Fonts](#) set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F2** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **mono-spaced bold**. For example:

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System → Preferences → Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications → Accessories → Character Map** from the main menu bar. Next, choose **Search → Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit → Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or Proportional Bold Italic

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable

text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount file-system** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q package** command. It will return a result as follows: **package-version-release**.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books      Desktop   documentation  drafts   mss      photos    stuff    svn
books_tests  Desktop1  downloads       images   notes    scripts   svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object ref = iniCtx.lookup("EchoBean");
        EchoHome home = (EchoHome) ref;
        Echo echo = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.



Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Getting Help and Giving Feedback

2.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at <http://access.redhat.com>. Through the customer portal, you can:

- ▶ search or browse through a knowledgebase of technical support articles about Red Hat products.
- ▶ submit a support case to Red Hat Global Support Services (GSS).
- ▶ access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at <https://www.redhat.com/mailman/listinfo>. Click on the name of any mailing list to subscribe to that list or to access the list archives.

2.2. Give us Feedback

If you find a typographical error, or know how this guide can be improved, we would love to hear from you. Submit a report in Bugzilla against the product **JBoss Enterprise Application Platform 6** and the component **doc-Administration_and_Configuration_Guide**. The following link will take you to a pre-filled bug report for this product: <https://bugzilla.redhat.com/>.

Fill out the following template in Bugzilla's **Description** field. Be as specific as possible when describing the issue; this will help ensure that we can fix it quickly.

Document URL:

Section Number and Name:

Describe the issue:

Suggestions for improvement:

Additional information:

Be sure to give us your name so that you can receive full credit for reporting the issue.

Chapter 1. Introduction to Administering the JBoss Enterprise Application Platform

1.1. Introducing JBoss Enterprise Application Platform 6

JBoss Enterprise Application Platform 6 is a middleware platform built on open standards, and compliant with Java EE. It integrates JBoss Application Server 7 with high-availability clustering, powerful messaging, distributed caching, and other technologies to create a stable, scalable, and fast platform. In addition, it also includes APIs and development frameworks you can use to develop secure, powerful, and scalable Java EE applications quickly.

[Report a bug](#)

1.2. New and Changed Features in JBoss Enterprise Application Platform 6

- ▶ JBoss Enterprise Application Platform 6 is a certified implementation of the Java Enterprise Edition 6 Full Profile and Web Profile specifications.
- ▶ A Managed Domain provides centralized management of multiple server instances and physical hosts, while a Standalone Server allows for a single server instance.
- ▶ Configurations, deployments, socket bindings, modules, extensions, and system properties can all be managed per server group.
- ▶ The Management Console and Management CLI are brand new interfaces for managing your domain or standalone JBoss Enterprise Application Platform 6 instance. There is no longer any need to edit XML configuration files by hand. The Management CLI even offers batch mode, so that you can script and automate management tasks.
- ▶ Application security, including security domains, are managed centrally for simplified configuration.
- ▶ The directory layout of JBoss Enterprise Application Platform 6 has been simplified. The **modules/** directory now contains the application server modules, instead of using common and server-specific **lib/** directories. The **domain/** and **standalone/** directories contain the artifacts and configuration files for domain and standalone deployments.
- ▶ The classloading mechanism has been made completely modular, so that modules are loaded and unloaded on demand. This provides performance and security benefits, as well as very fast start-up and restart times.
- ▶ Datasource management is streamlined. Database drivers can be deployed just like other services. In addition, datasources are created and managed directly in the Management Console or Management CLI.
- ▶ JBoss Enterprise Application Platform 6 starts and stops very quickly, which is especially beneficial to developers. It uses fewer resources and is extremely efficient in its use of system resources.

[Report a bug](#)

Chapter 2. Application Server Management

2.1. Manage the Application Server

JBoss Enterprise Application Platform 6 offers you multiple management tools to configure and administer your implementation as you require. These include the new Management Console or the Management Command Line Interface (CLI), as examples of the underlying Management API that enables expert users to develop their own tools if they desire.

[Report a bug](#)

2.2. Installation Structure and Details

JBoss Enterprise Application Platform 6 includes a simplified directory structure, compared to previous versions. Following is a listing of the directory structure, and a description of what the directory contains.

Table 2.1. Top-level directories and files

Name	Purpose
appclient/	Contains configuration details for the application client container.
bin/	Contains start-up scripts for JBoss Enterprise Application Platform 6 on Red Hat Enterprise Linux and Microsoft Windows.
bundles/	Contains OSGi bundles which pertain to JBoss Enterprise Application Platform 6 internal functionality.
docs/	License files, schemas, and examples.
domain/	Configuration files, deployment content, and writable areas used when JBoss Enterprise Application Platform 6 runs as a managed domain.
modules/	Modules which are dynamically loaded by JBoss Enterprise Application Platform 6 when services request them.
standalone/	Configuration files, deployment content, and writable areas used when JBoss Enterprise Application Platform 6 runs as a standalone server.
welcome-content/	Contains content used by the Welcome web application which is available on port 8080 of a default installation.
jboss-modules.jar	The bootstrapping mechanism which loads modules.

Table 2.2. Directories within the domain/ directory

Name	Purpose
configuration/	Configuration files for the managed domain. These files are modified by the Management Console and Management CLI, and are not meant to be edited directly.
data/	Information about deployed services. Services are deployed using the Management Console and Management CLI, rather than by a deployment scanner. Therefore, do not place files in this directory manually.
log/	Contains the run-time log files for the host and process controllers which run on the local instance.
servers/	Contains the equivalent data/ , log/ , and tmp/ directories for each server instance in a domain, which contain similar data to the same directories within the top-level domain/ directory.
tmp/	Contains temporary data such as files pertaining to the shared-key mechanism used by the Management CLI to authenticate local users to the managed domain.

Table 2.3. Directories within the standalone/ directory

Name	Purpose
configuration/	Configuration files for the standalone server. These files are modified by the Management Console and Management CLI, and are not meant to be edited directly.
deployments/	Information about deployed services. The standalone server does include a deployment scanner, so you can place archives in this directory to be deployed. However, the recommended approach is to manage deployments using the Management Console or Management CLI.
lib/	External libraries which pertain to a standalone server mode. Empty by default.
tmp/	Contains temporary data such as files pertaining to the shared-key mechanism used by the Management CLI to authenticate local users to the server.

[Report a bug](#)

2.3. JBoss Enterprise Application Platform 6 Profiles

The concept of profiles that was used in previous versions of JBoss Enterprise Application Platform is no longer used. JBoss Enterprise Application Platform 6 now uses a small number of configuration files to hold all information about its configuration.

Modules and drivers are loaded on an as-needed basis, so the concept of a default profile which was used in previous versions of JBoss Enterprise Application Platform 6, where profiles were used to make the server start more efficiently, does not apply. At deployment time, module dependencies are determined, ordered, and resolved by the server or domain controller, and loaded in the correct order. During undeployment, modules are unloaded when no deployment needs them any longer.

It is possible to disable modules or undeploy drivers or other services manually by removing the subsystems from the configuration. However, for most cases this is unnecessary. If none of your applications use a module, it will not be loaded.

[Report a bug](#)

2.4. About JBoss Enterprise Application Platform 6 Configuration Files

The configuration for JBoss Enterprise Application Platform 6 has changed considerably from previous versions. One of the most obvious differences is the use of a simplified configuration file structure, which includes one or more of the files listed below.

Table 2.4. Configuration File Locations

Server mode	Location	Purpose
domain.xml	EAP_HOME/domain/configuration/domain.xml	This is the main configuration file for a managed domain. Only the domain master reads this file. On other domain members, it can be removed.
host.xml	EAP_HOME/domain/configuration/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. This file is not present for standalone servers.
host-master.xml	EAP_HOME/domain/configuration/host-master.xml	This file includes only the configuration details necessary to run a server as a managed domain master server. This file is not present for standalone servers.
host-slave.xml	EAP_HOME/domain/configuration/host-slave.xml	This file includes only the configuration details necessary to run a server as a managed domain slave server. This file is not present for standalone servers.
standalone.xml	EAP_HOME/standalone/configuration/standalone.xml	This is the default configuration file for a standalone server. It contains all information about the standalone server, including subsystems, networking, deployments, socket bindings, and other configurable details.
standalone-ha.xml	EAP_HOME/standalone/configuration/standalone-ha.xml	This configuration file enables the mod_cluster and JGroups subsystems for a standalone server, so that it can participate in a high-availability or load-balancing cluster. This file is not necessary for a managed domain.

These are only the default locations. You can specify a different configuration file at run-time.

[Report a bug](#)

2.5. Management APIs

2.5.1. Management Application Programming Interfaces (APIs)

Management clients

JBoss Enterprise Application Platform 6 offers three different approaches to configure and manage servers, being a web interface, a command line client and a set of XML configuration files. While the recommended methods for editing the configuration file include the Management Console and Management CLI, edits made to the configuration by all three are always synchronized across the different views and finally persisted to the XML files. Note that edits made to the XML configuration files while a server instance is running will be overwritten by the server model.

HTTP API

The Management Console is an example of a web interface built with the Google Web Toolkit (GWT). The Management Console communicates with the server using the HTTP management interface. The HTTP API endpoint is the entry point for management clients that rely on the HTTP protocol to integrate with the management layer. It uses a JSON encoded protocol and a de-typed, RPC-style API to describe and execute management operations against a Managed Domain or Standalone Server. The HTTP API is used by the web console, but offers integration capabilities for a wide range of other clients too.

The HTTP API endpoint is co-located with either the domain controller or a Standalone Server instance. The HTTP API Endpoint serves two different contexts; one for executing management operations and the other to access the web interface. By default, it runs on port 9990.

Example 2.1. HTTP API Configuration File Example

```
<management-interfaces>
  [...]
  <http-interface interface="management" port="9990"/>
</management-interfaces>
```

The web console is served through the same port as the HTTP management API. It is important to distinguish between the Management Console accessed as on a default localhost, the Management Console as accessed remotely by a specific host and port combination, and the exposed domain API.

Table 2.5. TableTitle

URL	Description
http://localhost:9990/console	The Management Console accessed on the local host, controlling the Managed Domain configuration.
http://hostname:9990/console	The Management Console accessed remotely, naming the host and controlling the Managed Domain configuration.
http://hostname:9990/management	The HTTP Management API runs on the same port as the Management Console, displaying the raw attributes and values exposed to the API.

Native API

An example of a Native API tool is the Management CLI. This management tool is available for a Managed Domain or Standalone Server instance, allowing the a user to connect to the domain controller or a Standalone Server instance and execute management operations available through the de-typed management model.

The Native API endpoint is the entry point for management clients that rely on the native protocol to integrate with the management layer. It uses an open binary protocol and an RPC-style API based on a very small number of Java types to describe and execute management operations. It's used by the Management CLI management tool, but offers integration capabilities for a wide range of other clients too.

The Native API endpoint is co-located with either a host controller or a Standalone Server. It must be enabled to use the Management CLI. By default, it runs on port 9999.

Example 2.2. Native API Configuration File Example

```
<management-interfaces>
  <native-interface interface="management" port="9999"/>
  [...]
<management-interfaces>
```

[Report a bug](#)

2.6. Start JBoss Enterprise Application Platform 6

2.6.1. Start JBoss Enterprise Application Platform 6

Task

Start JBoss Enterprise Application Platform 6 in one of the following ways:

- ▶ [Section 2.6.3, "Start JBoss Enterprise Application Platform 6 as a Managed Domain"](#)
- ▶ [Section 2.6.2, "Start JBoss Enterprise Application Platform 6 as a Standalone Server"](#)
- ▶ [Section 2.7.1, "Run JBoss Enterprise Application Platform 6 as an Operating System Service"](#)

[Report a bug](#)

2.6.2. Start JBoss Enterprise Application Platform 6 as a Standalone Server

Red Hat Enterprise Linux.

Run the command: `EAP_HOME/bin/standalone.sh`

Microsoft Windows Server.

Run the command: `EAP_HOME\bin\standalone.bat`

Optional: Specify additional parameters.

To print a list of additional parameters to pass to the start-up scripts, use the `-h` parameter.

Result

The JBoss Enterprise Application Platform 6 Standalone Server instance starts.

[Report a bug](#)

2.6.3. Start JBoss Enterprise Application Platform 6 as a Managed Domain

Red Hat Enterprise Linux.

Run the command: `EAP_HOME/bin/domain.sh`

Microsoft Windows Server.

Run the command: `EAP_HOME\bin\domain.bat`

Optional: Pass additional parameters to the start-up script

For a list of parameters you can pass to the start-up script, use the `-h` parameter.

Result

The JBoss Enterprise Application Platform 6 Managed Domain instance starts.

[Report a bug](#)

2.6.4. Start JBoss Enterprise Application Platform 6 with an Alternative Configuration

Task Summary

If you do not specify a configuration file, the server starts with the default file. However, when you start the server, you can specify a configuration manually. The process varies slightly, depending on whether you are using a Managed Domain or Standalone Server, and depending on which operating system you are using.

Task Prerequisites

Before using an alternate configuration file, prepare it using the default configuration as a template. For a Managed Domain, the configuration file needs to be placed in **EAP_HOME/domain/configuration/**. For a Standalone Server, the configuration file should be placed in **EAP_HOME/standalone/configuration/**.



Example configurations

Several example configurations are included in the configuration directories. Use these examples to enable extra features such as clustering or the Transactions XTS API.

1. Managed Domain

For a Managed Domain, provide the file name of the configuration file as an option to the **--domain-config** parameter. You do not need to give the full path, if the configuration file resides in the **EAP_HOME/domain/configuration/** directory.

Example 2.3. Using an alternate configuration file for a Managed Domain in Red Hat Enterprise Linux

```
[user@host bin]$ ./domain.sh --domain-config=domain-alternate.xml
```

Example 2.4. Using an alternate configuration file for a Managed Domain in Microsoft Windows Server

```
C:\EAP_HOME\bin> domain.bat --domain-config=domain-alternate.xml
```

2. Standalone server

For a Standalone Server, provide the filename of the configuration file as an option to the **--server-config** parameter. You do not need to give the full path to the configuration file if it is in the **EAP_HOME/standalone/configuration/** directory.

Example 2.5. Using an alternate configuration file for a Standalone Server in Red Hat Enterprise Linux

```
[user@host bin]$ ./standalone.sh --server-config=standalone-alternate.xml
```

Example 2.6. Using an alternate configuration file for a Standalone Server in Microsoft Windows Server

```
C:\EAP_HOME\bin> standalone.bat --server-config=standalone-alternate.xml
```

Result:

JBoss Enterprise Application Platform 6 is now running, using your alternate configuration.

[Report a bug](#)

2.6.5. Stop JBoss Enterprise Application Platform 6

Task Summary:

The way that you stop JBoss Enterprise Application Platform 6 depends on how it was started. This task covers stopping an instance that was started interactively, stopping an instance that was started by a service, and stopping an instance that was forked into the background by a script.



Note

This task does not address stopping a server or server group in a Managed Domain. For those scenarios, see [Section 2.8.3. "Stop a Server Using the Management Console"](#).

Procedure 2.1. Task:

1. Stop an instance which was started interactively from a command prompt.

Press **Ctrl+C** in the terminal where JBoss Enterprise Application Platform 6 is running.

2. Stop an instance which was started as an operating system service.

Depending on your operating system, use one of the following procedures.

A. Red Hat Enterprise Linux

For Red Hat Enterprise Linux, if you have written a service script, use its **stop** facility. This needs to be written into the script. Then you can use **service scriptname stop**, where **scriptname** is the name of your script.

B. Microsoft Windows Server

In Microsoft Windows, use the **net service** command, or stop the service from the **Services** applet in the Control Panel.

3. Stop an instance which is running in the background (Red Hat Enterprise Linux)

a. Locate the instance from the process list. One option is to run the command **ps aux |grep "[j]ava -server"**. This returns one result for each JBoss Enterprise Application Platform 6 instance that is running on the local machine.

b. Send the process the **TERM** signal, by running **kill process_ID**, where **process_ID** is the number in the second field of the **ps aux** command above.

Result:

Each of these alternatives shuts JBoss Enterprise Application Platform 6 down cleanly so that data is not lost.

[Report a bug](#)

2.6.6. Reference of Switches and Arguments to pass at Server Runtime

The application server startup script accepts the addition of arguments and switches at runtime. The use of these parameters allows for the server to be started under alternative configurations to those defined in the **standalone.xml**, **domain.xml** and **host.xml** configuration files. This might include starting the server with an alternative set of socket bindings or a secondary configuration. A list of these available parameters can be accessed by passing the help switch at startup.

Example 2.7.

The following example is similar to the server startup explained in [Section 2.6.2. "Start JBoss Enterprise Application Platform 6 as a Standalone Server"](#), with the addition of the **-h** or **--help** switches. The results of the help switch are explained in the table below.

```
[localhost bin]$ standalone.sh -h
```

Table 2.6. Table of runtime switches and arguments

Argument or Switch	Description
--admin-only	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.
-b=<value>	Set system property jboss.bind.address to the given value.
-b <value>	Set system property jboss.bind.address to the given value.
-b<interface>=<value>	Set system property jboss.bind.address.<interface> to the given value.
-c=<config>	Name of the server configuration file to use. The default is standalone.xml .
-c <config>	Name of the server configuration file to use. The default is standalone.xml .
-D<name>[=<value>]	Set a system property.
-h	Display the help message and exit.
--help	Display the help message and exit.
-P=<url>	Load system properties from the given URL.
-P <url>	Load system properties from the given URL.
--properties=<url>	Load system properties from the given URL.
-S<name>[=<value>]	Set a security property.
--server-config=<config>	Name of the server configuration file to use. The default is standalone.xml .
-u=<value>	Set system property jboss.default.multicast.address to the given value.
-u <value>	Set system property jboss.default.multicast.address to the given value.
-V	Display the application server version and exit.
-v	Display the application server version and exit.
--version	Display the application server version and exit.

[Report a bug](#)

2.7. Run JBoss Enterprise Application Platform 6 as a Service

2.7.1. Run JBoss Enterprise Application Platform 6 as an Operating System Service

JBoss Enterprise Application Platform 6 can be configured to run as a service, allowing you to start a Managed Domain or Standalone Server configuration at system runtime, and allowing the server instance to continue to run when you log out of your local system.

- ▶ [Section 2.7.2, “Install JBoss Enterprise Application Platform as a Service in Red Hat Enterprise Linux”](#)
- ▶ [Section 2.7.3, “Install JBoss Enterprise Application Platform 6 as a Service in Microsoft Windows”](#)

[Report a bug](#)

2.7.2. Install JBoss Enterprise Application Platform as a Service in Red Hat Enterprise Linux

Summary

Use the following procedure to install JBoss Enterprise Application Platform 6 as a service on Red Hat Enterprise Linux.

Prerequisites

You need administrator access to complete this task.

Procedure 2.2. Task

1. Copy the start-up script to the /etc/init.d/ directory

The start-up script and an associated configuration file are located in the **EAP_HOME/bin/init.d/** directory. Copy each of these files to the **/etc/init.d/** directory.

```
[user@host init.d]$ sudo cp jboss-as-standalone.sh jboss-as.conf  
/etc/init.d
```

2. Add the start-up script as a service.

Add the new **jboss-as-standalone.sh** service to list of automatically started services, using the **chkconfig** service management command.

```
[user@host init.d]$ sudo chkconfig --add jboss-as-standalone.sh
```

3. Edit the script options.

If desired, edit the **jboss-as.conf** file to customize start-up options for JBoss Enterprise Application Platform and the JVM. Use the comments in the file as guidance. It is recommended to set the **JBOSS_HOME** variable in this file, to point to the directory where you extracted JBoss Enterprise Application Platform 6. Do not add a trailing slash (/) at the end of the directory name.

4. Edit the script itself.

You may need to edit the start-up script itself. It makes certain assumptions about the name of your start-up file and the location of your JBoss Enterprise Application Platform instance. Customize the script, paying special attention to the following variables, which you will need to customize to start JBoss Enterprise Application Platform 6 as a managed domain.

- » **JBOSS_HOME** - the location where JBoss Enterprise Application Platform 6 is extracted
- » **JBOSS_USER** - the user with the ability to run JBoss Enterprise Application Platform. This should be a non-privileged user, as no superuser privileges are required.
- » **JBOSS_CONFIG** - the name of the configuration file used to start JBoss Enterprise Application Platform 6, such as **domain.xml** or **standalone.xml**
- » **JBOSS_SCRIPT** - the script used to start JBoss Enterprise Application Platform 6, such as **domain.sh** or **standalone.sh**

5. Start the service.

If desired, start the new service using the standard syntax for starting Red Hat Enterprise Linux services.

```
[user@host bin]$ sudo service jboss-as-standalone.sh start
```

Result

JBoss Enterprise Application Platform 6 starts automatically when the Red Hat Enterprise Linux reaches its default run-level, and stops automatically when the operating system goes through its shutdown routine.

[Report a bug](#)

2.7.3. Install JBoss Enterprise Application Platform 6 as a Service in Microsoft Windows

Summary

This task installs JBoss Enterprise Application Platform 6 as a service on Microsoft Windows.

Prerequisites

You need administrator access to complete this task.

Procedure 2.3. Task

1. Download the Native Utilities package for your architecture.

32-bit, 64-bit, and Itanium 64-bit packages are available from the Red Hat Customer Portal at <https://access.redhat.com>. For more information on downloading software from the Red Hat Customer Portal, refer to the *JBoss Enterprise Application Platform 6 Installation Guide*, available here: https://access.redhat.com/knowledge/docs/JBoss_Enterprise_Application_Platform/.

2. Unzip the downloaded archive.

Unzip the archive into a new folder.

Result: The modules\native\bin\ folder is created.

The **modules\native\bin** folder contains the files you need to install JBoss Enterprise Application Platform 6 as a service. These services are part of *Procrun*, which is a series of wrapper scripts provided by Apache Commons. To learn more about *Procrun* and its syntax, refer to the following link: <http://commons.apache.org/daemon/procrun.html>.

3. Run the modules\sbin\prunsrv.exe executable.

```
prunsrv.exe install path_to_startup_script
```

Result

The service is installed. JBoss Enterprise Application Platform 6 is listed in the Services applet **services.msc**.

4. Manage your service.

Use the **modules\bin\prunmgr.exe** executable to manage, edit, add, or delete services. The following command-line options are supported:

- ▶ run
- ▶ service
- ▶ start
- ▶ stop
- ▶ update
- ▶ install
- ▶ delete
- ▶ pause [seconds]
- ▶ version
- ▶ help

The general syntax is:

```
prunmgr.exe command service_name
```

Result

You can use the **net service** command at the command line, or the **services.msc** applet, to start, stop, and manage automatic start-up of JBoss Enterprise Application Platform 6 in Microsoft Windows Server.

[Report a bug](#)

2.8. Start and Stop Servers

2.8.1. Start and Stop Servers

You can start and stop servers with the Management CLI or the Management Console.

If you are running a Standalone Server instance, you can shut the server down with the **shutdown** operation in the Management CLI. There is no specific equivalent in the Management Console, as you are free to use your filesystem to shut down the running instance.

If you are running a Managed Domain, the Management Console allows you to selectively start or stop specific servers in the domain. The Management CLI allows you to start all inactive servers, and stop any servers currently running. Like with the Standalone Server instance, the **shutdown** operation will shut down the server, in this case specifically the domain controller, all host controllers and their server instances.

[Report a bug](#)

2.8.2. Start a Server Using the Management Console

Prerequisites

- ▶ [Section 2.6.3, "Start JBoss Enterprise Application Platform 6 as a Managed Domain"](#)
- ▶ [Section 3.2.2, "Log in to the Management Console"](#)

Procedure 2.4. Task

1. Navigate to Server Instances in the Management Console

- Select the **Runtime** tab from the top-right of the console.
- Select **Domain Status** → **Server Instances** from the menu on the left of the console.

The screenshot shows the JBoss Enterprise Application Platform 6.0 Management Console. The title bar reads "JBoss' ENTERPRISE APPLICATION PLATFORM 6.0". The top navigation bar has tabs for "Profiles", "Server", and "Runtime", with "Runtime" being the active tab. A message indicator "(2) Messages" is shown in the top right. On the left, a sidebar menu includes "Server" (set to "master: server-one"), "Domain Status" (expanded), "Server Instances" (selected), "JVM Status", "Subsystem Metrics" (expanded), "Datasources", "JPA", "JMS Destinations", "Transactions", "Web", "Runtime Operations" (expanded), "OSGi", and "Deployments" (expanded). Under Deployments, there are links for "Manage Deployments" and "Webservices". The main content area is titled "Server Status (Host: master)" and contains a sub-section "Server Instances represent the server runtime state. This includes the virtual machine status, as well as deployments and subsystem specific state (i.e. datasource pool sizes)". Below this is a table titled "Server Instances" with columns "Server", "Server Group", "Status", and "Active". It lists three servers: "server-one" (main-server-group, Active), "server-three" (other-server-group, Inactive), and "server-two" (main-server-group, Inactive). A "Start" button is located at the top right of the table. At the bottom of the table are navigation buttons (« « « » » ») and the text "1-3 of 3". Below the table, there are sections for "Status" (with "Availability" selected) and "Deployments" (with "Server Instance: server-two" and "Server Configuration: server-two"). Under "Deployments", it says "Running?: false".

Figure 2.1. Server Instances

2. Select a server

From the list of **Server Instances**, select the server you want to start. Servers that are running are indicated by a check mark.

3. Click the Start button

Click on the **Start** button above the server list to open the confirmation dialogue box. Click the **Confirm** button to start the server.

The screenshot shows a confirmation dialog box titled "Start Server Instance". The question inside the box is "Really modify Server Instance server-two?". Below the question, there are two buttons: "Confirm" (highlighted in blue) and "Cancel". The background of the dialog is semi-transparent, showing the "Server Instances" table from Figure 2.1. In the table, "server-one" and "server-three" are in their original states, while "server-two" is highlighted in light blue. The "Status" and "Availability" sections of the management console are also visible in the background.

Figure 2.2. Confirm server modification

Result

The selected server is started and running.

The screenshot shows the JBoss Enterprise Application Platform 6 Management Console. The left sidebar has a 'Server' dropdown set to 'master: server-one'. Under 'Server Instances', 'server-one' is selected. The main content area is titled 'Server Status (Host: master)'. It says 'Server instances represent the server runtime state. This includes the virtual machine status, as well as deployments and subsystem specific state (i.e. datasource pool sizes.)'. Below this is a table:

Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✗
server-two	main-server-group		✓

At the bottom of the table are navigation icons: back, forward, and page numbers (1-3 of 3). Below the table is a 'Status' section with a 'Availability' button. At the bottom, it shows 'Server Instance: server-two' and 'Server Configuration: server-two' with 'Running?: true'.

Figure 2.3. Started server[Report a bug](#)

2.8.3. Stop a Server Using the Management Console

Prerequisites

- ▶ [Section 2.6.3. "Start JBoss Enterprise Application Platform 6 as a Managed Domain"](#)
- ▶ [Section 3.2.2. "Log in to the Management Console"](#)

Procedure 2.5. Task

1. Navigate to Server Instances in the Management Console

- a. Select the **Runtime** tab from the top-right of the console.
- b. Select **Domain Status** → **Server Instances** from the menu on the left of the console.

The screenshot shows the JBoss Enterprise Application Platform 6 Management Console. The left sidebar has a 'Server' dropdown set to 'master: server-one'. Under 'Server Instances', 'server-one' is selected. The main content area is titled 'Server Status (Host: master)'. It says 'Server instances represent the server runtime state. This includes the virtual machine status, as well as deployments and subsystem specific state (i.e. datasource pool sizes.)'. Below this is a table:

Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✗
server-two	main-server-group		✗

At the bottom of the table are navigation icons: back, forward, and page numbers (1-3 of 3). Below the table is a 'Status' section with a 'Availability' button. At the bottom, it shows 'Server Instance: server-two' and 'Server Configuration: server-two' with 'Running?: false'.

Figure 2.4. Server Instances

2. Select a server

From the list of **Server Instances**, select the server you want to stop. Servers that are running are indicated by a check mark.

3. Click the Stop button

Click on the **Stop** button above the server list to open the confirmation dialogue box. Click the **Confirm** button to start the server.

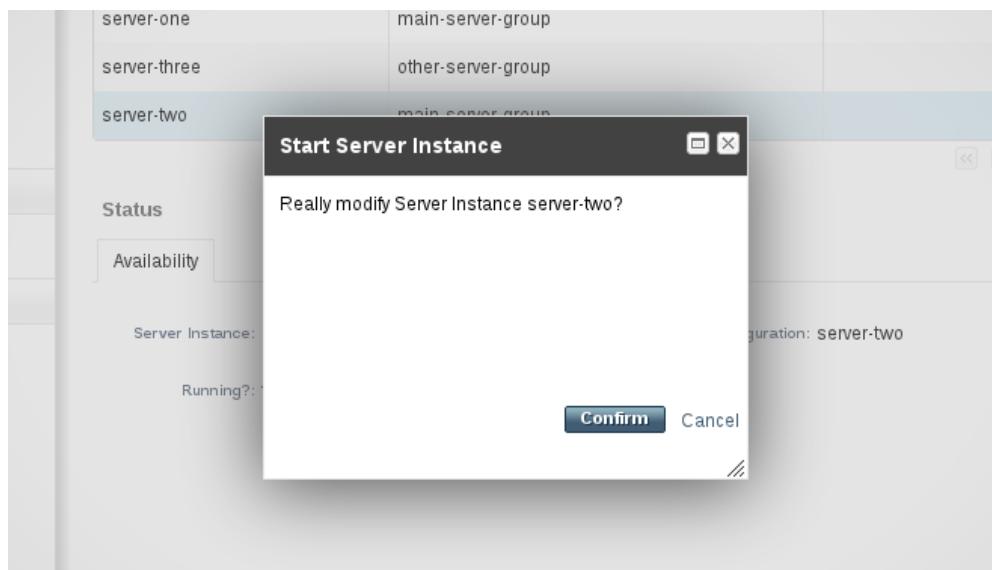


Figure 2.5. Confirm server modification

Result

The selected server is stopped.

Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✗
server-two	main-server-group		✗

Figure 2.6. Stopped server

[Report a bug](#)

2.9. Filesystem Paths

2.9.1. Filesystem Paths

JBoss Enterprise Application Platform 6 uses logical names for filesystem paths. The **domain.xml**, **host.xml** and **standalone.xml** configurations all include a section where paths can be declared. Other sections of the configuration can then reference those paths by their logical name, avoiding the declaration of the absolute path for each instance. This benefits configuration and administration efforts as it allows specific host configurations to resolve to universal logical names.

For example, the logging subsystem configuration includes a reference to the **jboss.server.log.dir** path that points to the server's **log** directory.

Example 2.8. Relative path example for the logging directory

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

JBoss Enterprise Application Platform 6 automatically provides a number of standard paths without any need for the user to configure them in a configuration file.

Table 2.7. Standard Paths

Value	Description
jboss.home	The root directory of the JBoss EAP 6 distribution.
user.home	The user home directory.
user.dir	The user's current working directory.
java.home	The Java installation directory
jboss.server.base	The root directory for an individual server instance.
.dir	
jboss.server.data	The directory the server will use for persistent data file storage.
.dir	
jboss.server.log	The directory the server will use for log file storage.
.dir	
jboss.server.tmp	The directory the server will use for temporary file storage.
.dir	
jboss.domain.servers.dir	The directory under which a host controller will create the working area for individual server instances in a managed domain.

Users can add their own paths or override all except the first five of the above by adding a **path** element to their configuration file. The following example shows a new relative path declaration relative to the root directory for the individual server instance.

Example 2.9. Format of a relative path

```
<path name="examplename" path="example/path" relative-to="jboss.server.data.dir"/>
```

The structure of a path declaration uses the following attributes.

Table 2.8. Path Attributes

Attribute	Description
name	The name of the path.
path	The actual filesystem path. 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	An optional attribute indicating the name of another previously named path, or of one of the standard paths provided by the system.

A **path** element in a **domain.xml** configuration file only requires the name attribute. It does not need to include any information indicating what the actual filesystem path is, as shown in the following example.

Example 2.10. Domain path example

```
<path name="example"/>
```

This configuration simply declares that there is a path named **example** that the other parts of the **domain.xml** configuration can reference. The actual filesystem location declared by **example** is specific to the respective **host.xml** configuration files of the host instances joining the domain groups. If this approach is used, there must be a path element in each machine's **host.xml** that specifies what the actual filesystem path is.

Example 2.11. Host path example

```
<path name="example" path="path/to/example" />
```

A **path** element in a **standalone.xml** must include the specification of the actual filesystem path.

[Report a bug](#)

2.10. Configuration File History

2.10.1. Configuration File History

The application server configuration files include the **standalone.xml** instance, as well as the **domain.xml** and **host.xml** files. While these files may be modified by direct editing, the recommended method is to configure the application server model with the available management operations, including the Management CLI and the Management Console.

To assist in the maintenance and management of the server instance, the application server creates a timestamped version of the original configuration file at the time of startup. Any additional configuration changes made by management operations result in the original file being automatically backed up, and a working copy of the instance being preserved for reference and rollback. This archival functionality extends to saving, loading and deleting snapshots of the server configuration to allow for recall and rollback scenarios.

- ▶ [Section 2.10.2, "Start the Server with a Previous Configuration"](#)
- ▶ [Section 2.10.3, "Save a Configuration Snapshot Using the Management CLI"](#)
- ▶ [Section 2.10.4, "Load a Configuration Snapshot"](#)
- ▶ [Section 2.10.5, "Delete a Configuration Snapshot Using Management CLI"](#)
- ▶ [Section 2.10.6, "List All Configuration Snapshots Using Management CLI"](#)

[Report a bug](#)

2.10.2. Start the Server with a Previous Configuration

The following example shows how to start the application server with a previous configuration in a standalone server with **standalone.xml**. The same concept applies to a managed domain with **domain.xml** and **host.xml** respectively.

This example recalls a previous configuration saved automatically by the application server as management operations modify the server model.

1. Identify the backed up version that you want to start. This example will recall the instance of the server model prior to the first modification after successfully booting up.

```
EAP_HOME/configuration/standalone_xml_history/current/standalone.v1.xml
```

2. Start the server with this instance of the backed up model by passing in the relative filename under **jboss.server.config.dir**.

```
EAP_HOME/bin/standalone.sh --server-
config=standalone_xml_history/current/standalone.v1.xml
```

Result

The application server starts with the selected configuration.

[Report a bug](#)

2.10.3. Save a Configuration Snapshot Using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Snapshots are a point-in-time copy of the current server instance. These copies can be saved and loaded by the administrator.

The following example uses the `standalone.xml` instance, but the same process applies to the `domain.xml` and `host.xml` models.

Task

- ▶ **Save a snapshot**

Run the `:take-snapshot` operation to capture a copy of the current server instance.

```
[standalone@localhost:9999 /] :take-snapshot
{
    "outcome" => "success",
    "result" =>
"/home/User/EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20
110630-172258657standalone.xml"
}
```

Result

A snapshot of the current server instance has been saved.

[Report a bug](#)

2.10.4. Load a Configuration Snapshot

Snapshots are a point-in-time copy of the current server instance. These copies can be saved and loaded by the administrator. The process of loading snapshots is similar to the method used to [Section 2.10.2, “Start the Server with a Previous Configuration”](#), running from the command line rather than the Management CLI interface used to create, list and delete snapshots.

The following example uses the `standalone.xml` instance, but the same process applies to the `domain.xml` and `host.xml` models.

Procedure 2.6. Task

1. Identify the snapshot to be loaded. This example will recall the following file from the snapshot directory. The default path for the snapshot files is as follows.

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/2011081
2-191301472standalone.xml
```

The snapshots are expressed by their relative paths, by which the above example can be written as follows.

```
jboss.server.config.dir/standalone_xml_history/snapshot/20110812-
191301472standalone.xml
```

2. Start the server with the selected snapshot instance by passing in the filename.

```
EAP_HOME/bin/standalone.sh --server-
config=standalone_xml_history/snapshot/20110913-164449522standalone.xml
```

Result

The server restarts with the selected snapshot profile.

[Report a bug](#)

2.10.5. Delete a Configuration Snapshot Using Management CLI

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)

Snapshots are a point-in-time copy of the current server instance. These copies can be saved and loaded by the administrator.

The following examples use the `standalone.xml` instance, but the same process applies to the `domain.xml` and `host.xml` models.

Procedure 2.7. Delete a Specific Snapshot

1. Identify the snapshot to be deleted. This example will delete the following file from the snapshot directory.

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110630-165714239standalone.xml
```

2. Run the `:delete-snapshot` command to delete a specific snapshot, specifying the name of the snapshot as in the example below.

```
[standalone@localhost:9999 /] :delete-snapshot(name="20110630-165714239standalone.xml")
{"outcome" => "success"}
```

Result

The snapshot has been deleted.

Procedure 2.8. Delete All Snapshots

- ▶ Run the `:delete-snapshot(name="all")` command to delete all snapshots as in the example below.

```
[standalone@localhost:9999 /] :delete-snapshot(name="all")
{"outcome" => "success"}
```

Result

All snapshots have been deleted.

[Report a bug](#)

2.10.6. List All Configuration Snapshots Using Management CLI

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)

Snapshots are a point-in-time copy of the current server instance. These copies can be saved and loaded by the administrator.

The following example uses the `standalone.xml` instance, but the same process applies to the `domain.xml` and `host.xml` models.

Procedure 2.9. Task

- ▶ **List all snapshots**

List all of the saved snapshots by running the `:list-snapshots` command.

```
[standalone@localhost:9999 /] :list-snapshots
{
    "outcome" => "success",
    "result" => {
        "directory" =>
        "/home/hostname/EAP_Home/standalone/configuration/standalone_xml_history/snapshot",
        "names" => [
            "20110818-133719699standalone.xml",
            "20110809-141225039standalone.xml",
            "20110802-152010683standalone.xml",
            "20110808-161118457standalone.xml",
            "20110912-151949212standalone.xml",
            "20110804-162951670standalone.xml"
        ]
    }
}
```

Result

The snapshots are listed.

[Report a bug](#)

Chapter 3. Management Interfaces

3.1. About the Management Console and Management CLI

In JBoss Enterprise Application Platform 6, all server instances and configurations are managed through management interfaces rather than by editing XML files. While the configuration XML files are still available for editing, administration through the management interfaces provides extra validation and advanced features for the persistent management of server instances. Changes made to the XML configuration files while the server instance is running will be overwritten by the server model, and any XML comments added will be removed as well. Only the management interfaces should be used for modifying the configuration files while a server instance is running.

To manage servers through a graphical user-interface in a web browser, use the Management Console.

To manage servers through a command line interface, use the Management CLI.

[Report a bug](#)

3.2. The Management Console

3.2.1. Management Console

The Management Console is a web-based administration tool for JBoss Enterprise Application Platform 6.

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 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.

[Report a bug](#)

3.2.2. Log in to the Management Console

Prerequisites

- » JBoss Enterprise Application Platform 6 must be running.

Procedure 3.1. Task

1. Navigate to the Management Console start page

Navigate to the Management Console in your web browser. The default location is <http://localhost:9990/console/>, where port 9990 is predefined as the Management Console socket binding.

2. Log in to the Management Console

Enter the username and password of the account that you created previously to log into the Management Console login screen.

The server localhost:9990 requires a username and password. The server says:
ManagementRealm.

User Name:

Password:

Figure 3.1. Log in screen for the Management Console

Result

Once logged in, one of the Management Console landing pages appears:

Managed domain

<http://localhost:9990/console/App.html#server-instances>

Standalone server

<http://localhost:9990/console/App.html#server-overview>

[Report a bug](#)

3.2.3. Change the Language of the Management Console

The language settings of web-based Management Console use English by default. You can choose to use one of the following languages instead.

Supported Languages

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

Procedure 3.2. Task

1. **Log into the Management Console.**

Log into the web-based Management Console.

2. **Open the Settings dialog.**

Near the bottom right of the screen is a **Settings** label. Click it to open the settings for the Management Console.

3. **Select the desired language.**

Select the desired language from the **Locale** selection box. Select **Save**. A confirmation box informs you that you need to reload the application. Click **Confirm**. Refresh your web browser to use the new locale.

[Report a bug](#)

3.2.4. Configure a Server Using the Management Console

Prerequisites

- » [Section 2.6.3, "Start JBoss Enterprise Application Platform 6 as a Managed Domain"](#)
- » [Section 3.2.2, "Log in to the Management Console"](#)

Procedure 3.3. Task

1. **Navigate to the server's Server Configuration panel in the Management Console**

- a. Select the **Server** tab from the top-right of the console.
- b. Expand the **Server Configurations** menu item on the left of the console and select the relevant server from the list.

Configuration Name	Server Group
server-one	main-server-group
server-three	other-server-group
server-two	main-server-group

Figure 3.2. Server configuration

2. Edit the server configuration

- Select the **Edit** button beneath the server list.
- Make changes to the configuration attributes.
- Select the **Save** button beneath the server list.

Result

The server configuration is changed, and will take effect next time the server restarts.

[Report a bug](#)

3.2.5. Add a Deployment in the Management Console

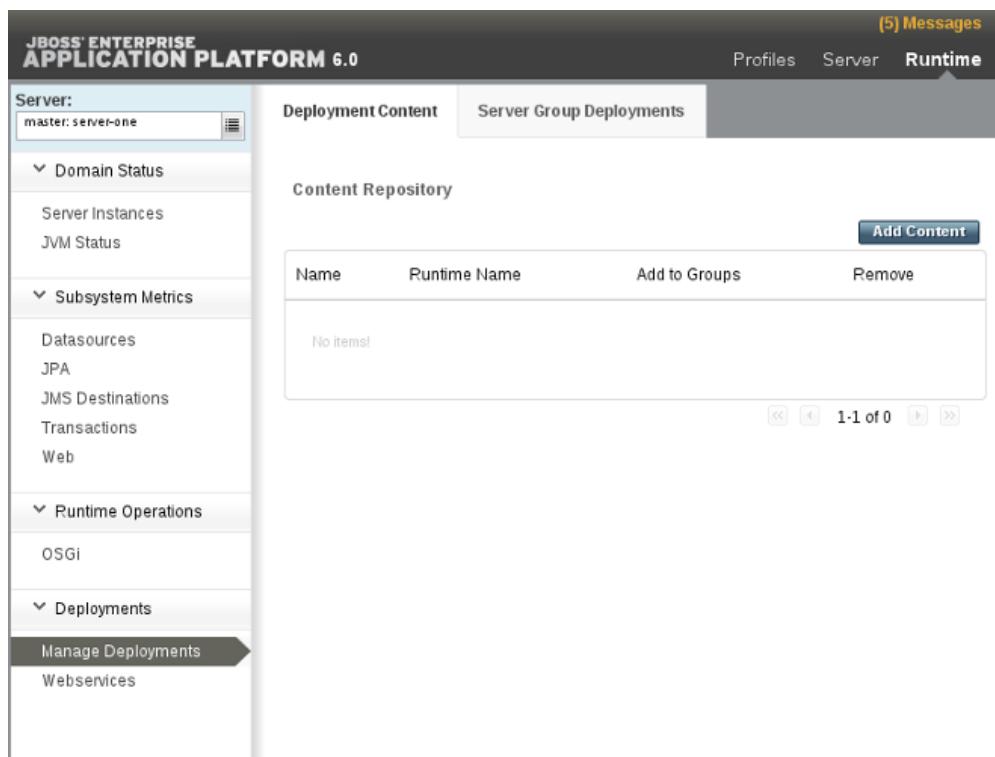
Prerequisites

- ▶ [Section 3.2.2. "Log in to the Management Console"](#)

Procedure 3.4. Task

1. **Navigate to the Manage Deployments panel in the Management Console**
 - Select the **Runtime** tab from the top right of the console.
 - For either a managed domain or a standalone server, select the **Deployments → Manage Deployments** option from the menu on the left of the console.

The **Manage Deployments** panel appears.

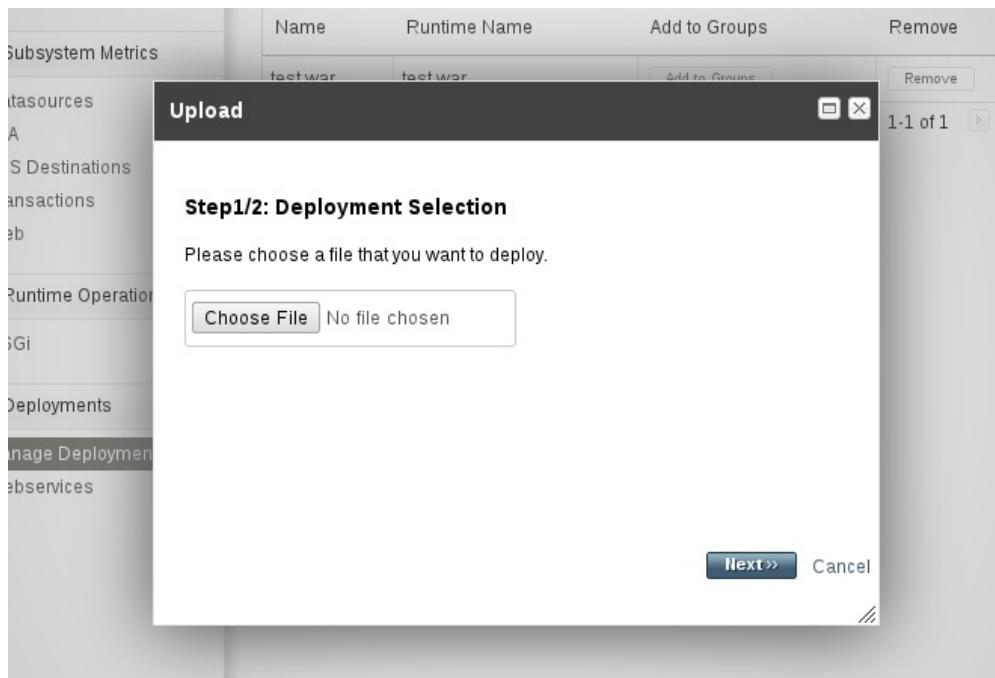
**Figure 3.3. Manage domain deployments**

2. Add deployment content

Select the **Add Content** button in the top right of the **Deployments** panel. An **Upload** dialog box appears.

3. Choose a file to deploy

In the dialog box, select the **Choose File** button. Browse to the file you want to deploy and select it for upload. Select the **Next** button to proceed once a file has been selected.

**Figure 3.4. Deployment selection**

4. Verify deployment names

Verify the deployment name and runtime name that appear in the **Upload** dialog box. Select the **Save** button to upload the file once the names are verified.

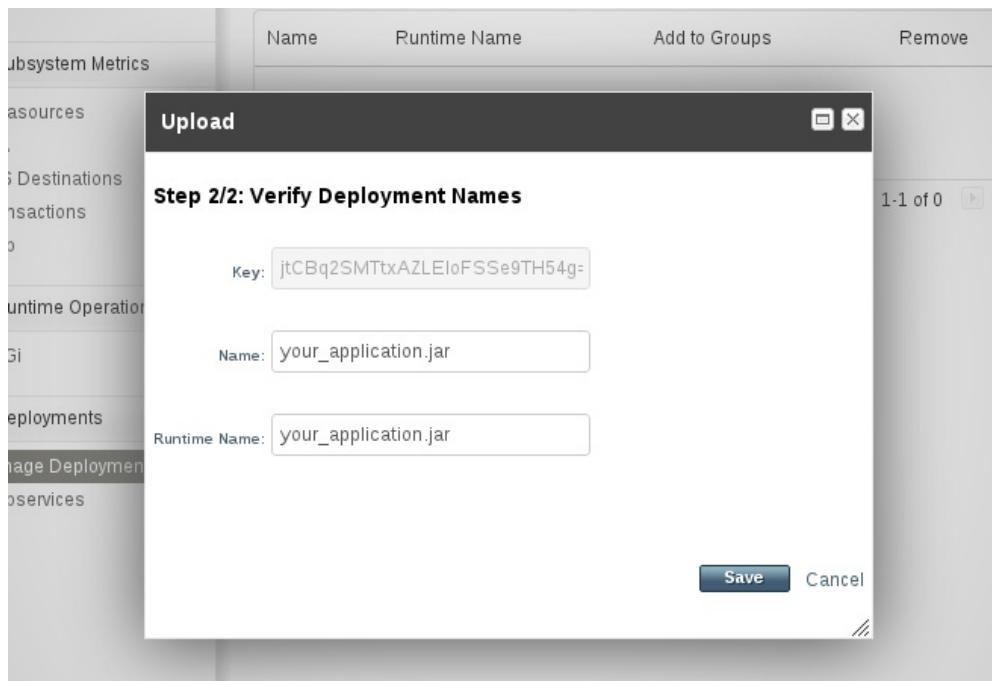


Figure 3.5. Verify deployment names

Result

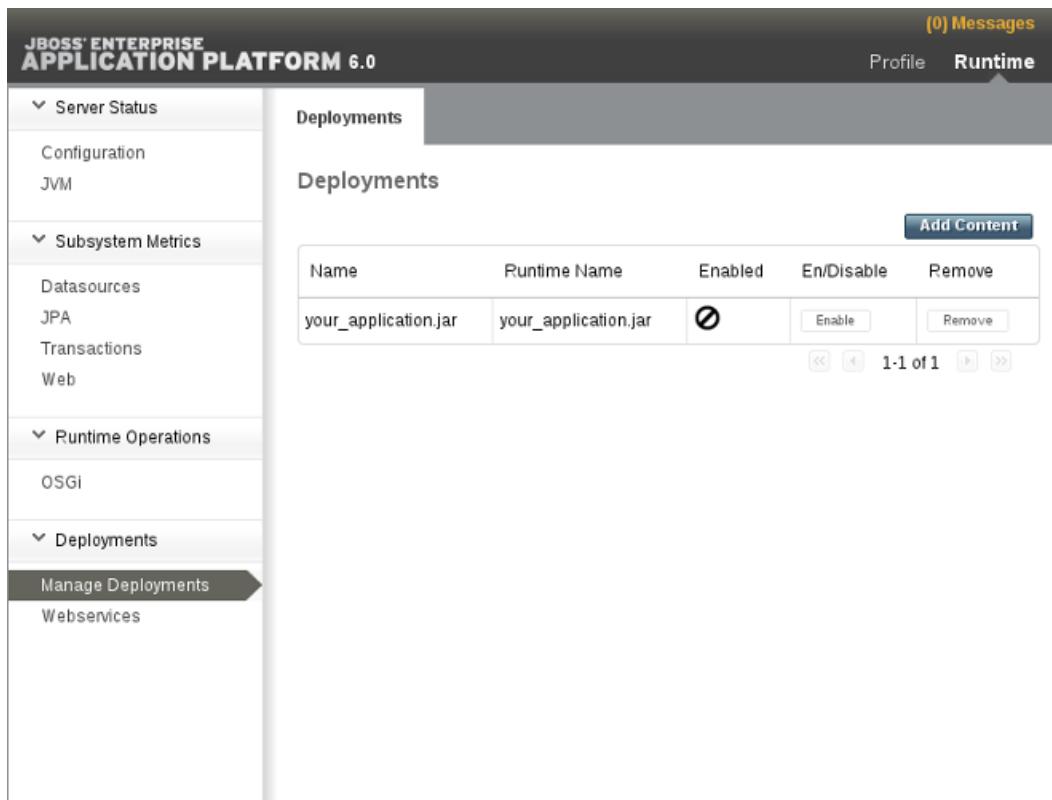
The selected content is uploaded to the server and is now ready for deployment.

The screenshot shows the 'Deployment Content' tab of the 'Content Repository' section. A table lists one deployment:

Name	Runtime Name	Add to Groups	Remove
your_application.jar	your_application.jar	Add to Groups	Remove

At the top right, there are tabs for 'Profiles', 'Server', and 'Runtime', with 'Runtime' being the active tab. The status bar at the bottom indicates '1-1 of 1'.

Figure 3.6. Uploaded deployment in a managed domain

**Figure 3.7. Uploaded deployment on a standalone server instance**[Report a bug](#)

3.2.6. Create a New Server in the Management Console

Prerequisites

- » [Section 2.6.3, "Start JBoss Enterprise Application Platform 6 as a Managed Domain"](#)
- » [Section 3.2.2, "Log in to the Management Console"](#)

Procedure 3.5. Task

1. Navigate to the Server Configurations page in the Management Console

Select the **Server** tab from the top-right of the console.

2. Create a new configuration

- a. Select the **Add** button at the top of the **Server Configuration** panel.
- b. Edit the basic server settings in the **Create Server Configuration** dialog.
- c. Select the **Save** button to save the new server configuration.

Result

The new server is created and listed in the **Server Configurations** list.

[Report a bug](#)

3.2.7. Change the Default Log Levels Using the Management Console

Procedure 3.6. Task

1. Navigate to the Logging panel in the Management Console

- a. If you are working with a managed domain, select the **Profiles** tab from the top-right of the console, then select the relevant profile from the drop-down list on the left of the console.
- b. For either a managed domain or a standalone server, select the **Core → Logging** option from the menu on the left of the console.
- c. Click on the **Log Categories** tab in the top of the console.

Name	Log Level
com.arjuna	WARN
jacob	WARN
jacob.config	ERROR
org.apache.tomcat.util.modeler	WARN

Figure 3.8. Logging panel

2. Edit logger details

Edit the details for any of the entries in the **Log Categories** table.

- Select an entry in the **Log Categories** table, then select the **Edit** button in the **Details** section below.
- Set the log level for the category with the **Log Level** drop-down box. Select the **Save** button when done.

Result

The log levels for the relevant categories are now updated.

[Report a bug](#)

3.2.8. Create a New Server Group in the Management Console

Prerequisites

- ▶ [Section 3.2.2, “Log in to the Management Console”](#)

Procedure 3.7. Task

1. Navigate to the Server Groups view

Select the **Profiles** tab in the top-right corner.

2. Select the Group Configurations tab under the Server Groups menu in the left hand column.

Group Configurations

Server Groups

A Server Group does specify a common management policy for a set of servers. Server Groups are associated with profiles.

Group Name	Profile
main-server-group	full
other-server-group	full-ha

Attributes JVM Configuration System Properties

Edit

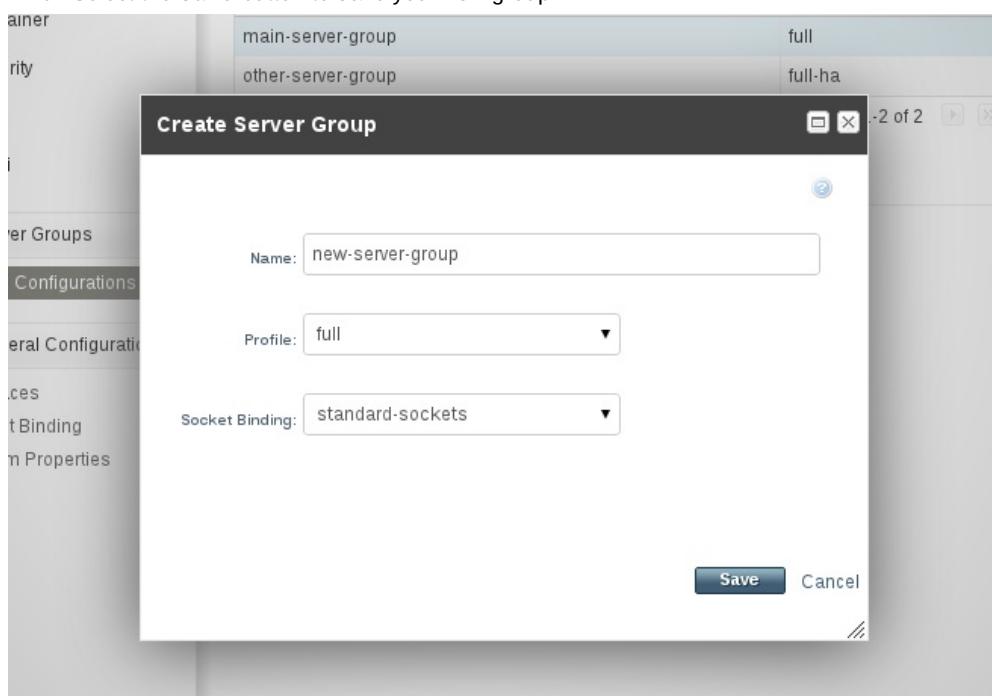
Name: main-server-group Profile: full
Socket Binding: standard-sockets

Figure 3.9. The Server Groups view**3. Add a server group**

Click the **Add** button to add a new server group.

4. Configure the server group

- Enter a name for the server group.
- Select the profile you want to add the server group to.
- Select the socket binding you want to add the server group to.
- Select the **Save** button to save your new group.

**Figure 3.10. The Create Server Group dialog**

Result

The new server group is visible in the Management Console.

[Report a bug](#)

3.3. The Management CLI

3.3.1. About the Management Command Line Interface (CLI)

The Management Command Line Interface (CLI) is a command line administration tool for JBoss Enterprise Application Platform 6.

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 modes, allowing multiple tasks to be run as a group.

[Report a bug](#)

3.3.2. Launch the Management CLI

Prerequisites

- ▶ Perform one of the following before accessing the Management CLI:
 - [Section 2.6.2, “Start JBoss Enterprise Application Platform 6 as a Standalone Server”](#)
 - [Section 2.6.3, “Start JBoss Enterprise Application Platform 6 as a Managed Domain”](#)

Procedure 3.8. Task

- ▶ A. **Launch the CLI in Linux**

Run the `EAP_HOME/bin/jboss-cli.sh` file by entering the following at a command line:

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

- ▶ B. **Launch the CLI in Windows**

Run the `EAP_HOME\bin\jboss-cli.bat` file by double-clicking it, or by entering the following at a command line:

```
C:\>EAP_HOME\bin\jboss-cli.bat
```

[Report a bug](#)

3.3.3. Quit the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Procedure 3.9. Task

- ▶ **Run the quit command**

From the Management CLI, enter the `quit` command:

```
[domain@localhost:9999 /] quit
Closed connection to localhost:9999
```

[Report a bug](#)

3.3.4. Connect to a Managed Server Instance Using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Procedure 3.10. Task

▶ **Run the connect command**

From the Management CLI, enter the **connect** command:

```
[disconnected /] connect
Connected to domain controller at localhost:9999
```

A. Alternatively, to connect to a managed server when starting the Management CLI on a Linux system, use the **--connect** parameter:

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

B. The **--connect** parameter can be used to specify the host and port of the server. To connect to the address **192.168.0.1** with the port value **9999** the following would apply:

```
$ EAP_HOME/bin/jboss-cli.sh --connect --controller=192.168.0.1:9999
```

[Report a bug](#)

3.3.5. Get Help with the Management CLI

Prerequisites

- ▶ [Section 3.3.2. "Launch the Management CLI"](#)

Summary

The Management CLI features a help dialog with general and context sensitive options. The help commands dependent on the operation context require an established connection to either a standalone or domain controller. These commands will not appear in the listing unless the connection has been established.

Procedure 3.11. Task

1. **Run the help command**

From the Management CLI, enter the **help** command:

```
[standalone@localhost:9999 /] help
```

2. **Get context sensitive help**

From the Management CLI, enter the **help -commands** extended command:

```
[standalone@localhost:9999 /] help --commands
```

3. For a more detailed description of a specific command, execute the **help** command with '**--help**' as the argument.

```
[standalone@localhost:9999 /] deploy --help
```

Result

The CLI help information is displayed.

[Report a bug](#)

3.3.6. Use the Management CLI in Batch Mode

Prerequisites

- ▶ [Section 3.3.2. "Launch the Management CLI"](#)
- ▶ [Section 3.3.4. "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 3.12. Task

Batch processing allows a number of operation requests to be grouped in a sequence and executed together as a unit. If any of the operation requests in the sequence fail, the entire group of operations is rolled back.

1. **Enter batch mode**

Enter batch mode with the **batch** command.

```
[standalone@localhost:9999 /] batch
[standalone@localhost:9999 / #]
```

Batch mode is indicated by the hash symbol (#) in the prompt.

2. Add operation requests to the batch

Once in batch mode, enter operation requests as normal. The operation requests are added to the batch in the order they are entered.

Refer to [Section 3.3.7, "Use Operations and Commands in the Management CLI"](#) for details on formatting operation requests.

3. Run the batch

Once the entire sequence of operation requests is entered, run the batch with the **run-batch** command.

```
[standalone@localhost:9999 / #] run-batch
The batch executed successfully.
```

Result

The entered sequence of operation requests is completed as a batch.

[Report a bug](#)

3.3.7. Use Operations and Commands in the Management CLI

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)
- ▶ [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 3.13. Task

1. Construct the operation request

Operation requests allow for low-level interaction with the management model. They provide a controlled way to edit server configurations. An operation request consists of three parts:

- ▶ an *address*, prefixed with a slash (/).
- ▶ an *operation name*, prefixed with a colon (:).
- ▶ an optional set of *parameters*, contained within parentheses (())�

a. Determine the address

The configuration is presented as a hierarchical tree of addressable resources. Each resource node offers a different set of operations. The address specifies which resource node to perform the operation on. An address uses the following syntax:

```
/node-type=node-name
```

- ▶ **node-type** is the resource node type. This maps to an element name in the configuration XML.
- ▶ **node-name** is the resource node name. This maps to the **name** attribute of the element in the configuration XML.
- ▶ Separate each level of the resource tree with a slash (/).

Refer to the configuration XML files to determine the required address. The

EAP_HOME/standalone/configuration/standalone.xml file holds the configuration for a standalone server and the

EAP_HOME/domain/configuration/domain.xml and

EAP_HOME/domain/configuration/host.xml files hold the configuration for a managed domain.

Example 3.1. Example operation addresses

To perform an operation on the logging subsystem, use the following address in an operation request:

```
/subsystem=logging
```

To perform an operation on the Java datasource, use the following address in an operation request:

```
/subsystem=datasources/data-source=java
```

b. Determine the operation

Operations differ for each different type of resource node. An operation uses the following syntax:

```
:operation-name
```

- » ***operation-name*** is the name of the operation to request.

Use the **read-operation-names** operation on any resource address in a standalone server to list the available operations.

Example 3.2. Available operations

To list all available operations for the logging subsystem, enter the following request for a standalone server:

```
[standalone@localhost:9999 /] /subsystem=logging:read-operation-names
{
    "outcome" => "success",
    "result" => [
        "add",
        "read-attribute",
        "read-children-names",
        "read-children-resources",
        "read-children-types",
        "read-operation-description",
        "read-operation-names",
        "read-resource",
        "read-resource-description",
        "remove",
        "undefine-attribute",
        "whoami",
        "write-attribute"
    ]
}
```

c. Determine any parameters

Each operation may require different parameters.

Parameters use the following syntax:

```
(parameter-name=parameter-value)
```

- » ***parameter-name*** is the name of the parameter.
- » ***parameter-value*** is the value of the parameter.
- » Multiple parameters are separated by commas (,).

To determine any required parameters, perform the **read-children-types** command on a resource node, passing the operation name as a parameter. Refer to [Example 3.3, “Determine operation parameters”](#) for details.

Example 3.3. Determine operation parameters

To determine any required parameters for the **read-children-types** operation on the logging subsystem, enter the **read-operation-description** command as follows:

```
[standalone@localhost:9999 /] /subsystem=logging:read-operation-
description(name=read-children-types)
{
    "outcome" => "success",
    "result" => {
        "operation-name" => "read-children-types",
        "description" => "Gets the type names of all the children
under the selected resource",
        "reply-properties" => {
            "type" => LIST,
            "description" => "The children types",
            "value-type" => STRING
        },
        "read-only" => false
    }
}
```

2. Enter the full operation request

Once the address, operation, and any parameters have been determined, enter the full operation request.

Example 3.4. Example operation request

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-
resource(recursive=true)
```

Result

The management interface performs the operation request on the server configuration.

[Report a bug](#)

3.3.8. Reference of Management CLI Commands**Prerequisites**

- » [Section 3.3.2. “Launch the Management CLI”](#)

Summary

The topic [Section 3.3.5, “Get Help with the Management CLI”](#) describes how to access the Management CLI help features, including a help dialogue with general and context sensitive options. The help commands are dependent on the operation context and require an established connection to either a standalone or domain controller. These commands will not appear in the listing unless the connection has been established.

Table 3.1.

Command	Description
batch	Starts the batch mode by creating a new batch or, depending on the existing held back batches, re-activates one. If there are no held back batches this command invoked w/o arguments will start a new batch. If there is an unnamed held back batch, this command will reactivate it. If there are named held back batches, they can be activated by executing this command with the name of the held back batch as the argument.
cd	Changes the current node path to the argument. The current node path is used as the address for operation requests that do not contain the address part. If an operation request does include the address, the included address is considered relative to the current node path. The current node path may end on a node-type. In that case, to execute an operation specifying a node-name would be sufficient, such as logging:read-resource.
clear	Clears the screen.
command	Allows you to add new, remove and list existing generic type commands. A generic type command is a command that is assigned to a specific node type and which allows you to perform any operation available for an instance of that type. It can also modify any of the properties exposed by the type on any existing instance.
connect	Connects to the controller on the specified host and port.
connection-factory	Defines a connection factory.
data-source	Manages JDBC datasource configurations in the datasource subsystem.
deploy	Deploys the application designated by the file path or enables an application that is pre-existing but disabled in the repository. If executed without arguments, this command will list all the existing deployments.
help	Displays the help message. Can be used with the --commands argument to provide context sensitive results for the given commands.
history	Displays the CLI command history in memory and displays a status of whether the history expansion is enabled or disabled. Can be used with arguments to clear, disable and enable the history expansion as required.
jms-queue	Defines a JMS queue in the messaging subsystem.
jms-topic	Defines a JMS topic in the messaging subsystem.
ls	List the contents of the node path. By default the result is printed in columns using the whole width of the terminal. Using the -l switch will print results on one name per line.
pwd	Prints the full node path of the current working node.
quit	Terminates the command line interface.
read-attribute	Prints the value and, depending on the arguments, the description of the attribute of a managed resource.
read-operation	Displays the description of a specified operation, or lists all available operations if none is specified.
undeploy	Undeploys an application when run with the name of the intended application. Can be run with arguments to remove the application from the repository also. Prints the list of all existing deployments when executed without an application specified.
version	Prints the application server version and environment information.
xa-data-source	Manages JDBC XA datasource configuration in the datasource subsystem.

[Report a bug](#)

3.3.9. Reference of Management CLI Operations

Exposing operations in the Management CLI

Operations in the Management CLI can be exposed by using the **read-operation-names** operation described in the topic [Section 3.4.5, “Display the Operation Names using the Management CLI”](#). The operation descriptions can be exposed by using the **read-operation-descriptions** operation described in the topic [Section 3.4.4, “Display an Operation Description using the Management CLI”](#).

Table 3.2. Management CLI operations

Operation Name	Description
add-namespace	Adds a namespace prefix mapping to the namespaces attribute's map.
add-schema-location	Adds a schema location mapping to the schema-locations attribute's map.
delete-snapshot	Deletes a snapshot of the server configuration from the snapshots directory.
full-replace-deployment	Add previously uploaded deployment content to the list of content available for use, replace existing content of the same name in the runtime, and remove the replaced content from the list of content available for use. Refer to link for further information.
list-snapshots	Lists the snapshots of the server configuration saved in the snapshots directory.
read-attribute	Displays the value of an attribute for the selected resource.
read-children-names	Displays the names of all children under the selected resource with the given type.
read-children-resources	Displays information about all of a resource's children that are of a given type.
read-children-types	Displays the type names of all the children under the selected resource.
read-config-as-xml	Reads the current configuration and displays it in XML format.
read-operation-description	Displays the details of an operation on the given resource.
read-operation-names	Displays the names of all the operations for the given resource.
read-resource	Displays a model resource's attribute values along with either basic or complete information about any child resources.
read-resource-description	Displays the description of a resource's attributes, types of children and operations.
reload	Reloads the server by shutting all services down and restarting.
remove-namespace	Removes a namespace prefix mapping from the namespaces attribute map.
remove-schema-location	Removes a schema location mapping from the schema-locations attribute map.
replace-deployment	Replace existing content in the runtime with new content. The new content must have been previously uploaded to the deployment content repository.
resolve-expression	Operation that accepts an expression as input or a string that can be parsed into an expression, and resolves it against the local system properties and environment variables.
resolve-internet-address	Takes a set of interface resolution criteria and finds an IP address on the local machine that matches the criteria, or fails if no matching IP address can be found.
server-set-restart-required	Puts the server into a restart-required mode
shutdown	Shuts down the server via a call to System.exit(0) .
start-servers	Starts all configured servers in a Managed Domain that are not currently running.
stop-servers	Stops all servers currently running in a Managed Domain.
take-snapshot	Takes a snapshot of the server configuration and saves it to the snapshots directory.
upload-deployment-bytes	Indicates that the deployment content in the included byte array should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.
upload-deployment-stream	Indicates that the deployment content available at the included input stream index should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.

upload-deployment-url	Indicates that the deployment content available at the included URL should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.
validate-address	Validates the operation's address.
write-attribute	Sets the value of an attribute for the selected resource.

[Report a bug](#)

3.4. Management CLI Operations

3.4.1. Display the Attributes of a Resource with the Management CLI

Prerequisites

- ▶ [Section 3.3.2. "Launch the Management CLI"](#)

Summary

The **read-attribute** operation is a global operation used to read the current runtime value of a selected attribute. It can be used to expose only the values that have been set by the user, ignoring any default or undefined values. The request properties include the following parameters.

Request Properties

name

The name of the attribute to get the value for under the selected resource.

include-defaults

A Boolean parameter that can be set to **false** to restrict the operation results to only show attributes set by the user and ignore default values.

Procedure 3.14. Task

- ▶ **Run the `read-attribute` operation**

From the Management CLI, use the **read-attribute** operation to display the value of a resource attribute. For more details on operation requests, refer to the topic [Section 3.3.7. "Use Operations and Commands in the Management CLI"](#).

```
[standalone@localhost:9999 /]:read-attribute(name=name-of-attribute)
```

An advantage of the **read-attribute** operation is the ability to expose the current runtime value of a specific attribute. Similar results can be achieved with the **read-resource** operation, but only with the addition of the **include-runtime** request property, and only as part of a list of all available resources for that node. The **read-attribute** operation is intended for fine-grained attribute queries, as the following example shows.

Example 3.5. Run the read-attribute operation to expose the public interface IP

If you know the name of the attribute that you would like to expose, you can use the **read-attribute** to return the exact value in the current runtime.

```
[standalone@localhost:9999 /] /interface=public:read-attribute(name=resolved-address)
{
    "outcome" => "success",
    "result" => "127.0.0.1"
}
```

The **resolved-address** attribute is a runtime value, so it is not displayed in the results of the standard **read-resource** operation.

```
[standalone@localhost:9999 /] /interface=public:read-resource
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
        "virtual" => undefined
    }
}
```

To display **resolved-address** and other runtime values, you must use the **include-runtime** request property.

```
[standalone@localhost:9999 /] /interface=public:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "resolved-address" => "127.0.0.1",
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
        "virtual" => undefined
    }
}
```

Result

The current runtime attribute value is displayed.

[Report a bug](#)

3.4.2. Display the Active User in the Management CLI

Prerequisites

- » [Section 3.3.2, "Launch the Management CLI"](#)

Summary

The **whoami** operation is a global operation used to identify the attributes of the current active user. The operation exposes the identity of the username and the realm that they are assigned to. The **whoami** operation is useful for administrators managing multiple users accounts across multiple realms, or to assist in keeping track of active users across domain instances with multiple terminal session and users accounts.

Procedure 3.15. Task

- » **Run the whoami operation**

From the Management CLI, use the **whoami** operation to display the active user account.

```
[standalone@localhost:9999 /] :whoami
```

The following example uses the **whoami** operation in a standalone server instance to show that the active user is **username**, and that the user is assigned to the **ManagementRealm** realm.

Example 3.6. Use the whoami in a standalone instance

```
[standalone@localhost:9999 /]:whoami
{
    "outcome" => "success",
    "result" => {"identity" => {
        "username" => "username",
        "realm" => "ManagementRealm"
    }}
}
```

Result

Your current active user account is displayed.

[Report a bug](#)

3.4.3. Display System and Server Information in the Management CLI

Prerequisites

- » [Section 3.3.2, "Launch the Management CLI"](#)

Procedure 3.16. Task

- » **Run the version command**

From the Management CLI, enter the **version** command:

```
[domain@localhost:9999 /] version
```

Result

Your application server version and environment information is displayed.

[Report a bug](#)

3.4.4. Display an Operation Description using the Management CLI

Prerequisites

- ▶ [Section 3.3.2. "Launch the Management CLI"](#)

Procedure 3.17. Task

- ▶ **Run the `read-operation-description` operation**

From the Management CLI, use **`read-operation-description`** to display information about the operation. The operation requires additional parameters in the format of a key-value pair to indicate which operation to display. For more details on operation requests, refer to the topic [Section 3.3.7. "Use Operations and Commands in the Management CLI"](#).

```
[standalone@localhost:9999 /]:read-operation-description(name=name-of-operation)
```

Example 3.7. Display the `list-snapshots` operation description

The following example shows the method for describing the **`list-snapshots`** operation.

```
[standalone@localhost:9999 /] :read-operation-description(name=list-snapshots)
{
    "outcome" => "success",
    "result" => {
        "operation-name" => "list-snapshots",
        "description" => "Lists the snapshots",
        "reply-properties" => {
            "type" => OBJECT,
            "value-type" => {
                "directory" => {
                    "type" => STRING,
                    "description" => "The directory where the snapshots are
stored"
                },
                "names" => {
                    "type" => LIST,
                    "value-type" => STRING,
                    "description" => "The names of the snapshots within the
snapshots directory"
                }
            },
            "read-only" => false
        }
    }
}
```

Result

The description is displayed for the chosen operation.

[Report a bug](#)

3.4.5. Display the Operation Names using the Management CLI

Prerequisites

- ▶ [Section 3.3.2. "Launch the Management CLI"](#)

Procedure 3.18. Task

- ▶ **Run the `read-operation-names` operation**

From the Management CLI, use the **`read-operation-names`** operation to display the names of the available operations. For more details on operation requests, refer to the topic [Section 3.3.7. "Use Operations and Commands in the Management CLI"](#).

```
[standalone@localhost:9999 /]:read-operation-names
```

Example 3.8. Display the operation names using the Management CLI

The following example shows the method for describing the **read-operation-names** operation.

```
[standalone@localhost:9999 /]:read-operation-names
{
    "outcome" => "success",
    "result" => [
        "add-namespace",
        "add-schema-location",
        "delete-snapshot",
        "full-replace-deployment",
        "list-snapshots",
        "read-attribute",
        "read-children-names",
        "read-children-resources",
        "read-children-types",
        "read-config-as-xml",
        "read-operation-description",
        "read-operation-names",
        "read-resource",
        "read-resource-description",
        "reload",
        "remove-namespace",
        "remove-schema-location",
        "replace-deployment",
        "shutdown",
        "take-snapshot",
        "upload-deployment-bytes",
        "upload-deployment-stream",
        "upload-deployment-url",
        "validate-address",
        "write-attribute"
    ]
}
```

Result

The available operation names are displayed.

[Report a bug](#)

3.4.6. Display Available Resources using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Summary

The **read-resource** operation is a global operation used to read resource values. It can be used to expose either basic or complete information about the resources of the current or child nodes, along with a range of request properties to expand or limit the scope of the operation results. The request properties include the following parameters.

Request Properties

recursive

Whether to recursively include complete information about child resources.

recursive-depth

The depth to which information about child resources should be included.

proxies

Whether to include remote resources in a recursive query. For example including the host level resources from slave Host Controllers in a query of the Domain Controller.

include-runtime

Whether to include runtime attributes in the response, such as attribute values that do not come from the persistent configuration. This request property is set to false by default.

include-defaults

A boolean request property that serves to enable or disable the reading of default attributes. When set to false only the attributes set by the user are returned, ignoring any that remain undefined.

Procedure 3.19. Task

1. Run the `read-resource` operation

From the Management CLI, use the **read-resource** operation to display the available resources.

```
[standalone@localhost:9999 /]:read-resource
```

The following example shows how you might use the **read-resource** operation on a standalone server instance to expose general resource information. The results resemble the **standalone.xml** configuration file, displaying the system resources, extensions, interfaces and subsystems installed or configured for the server instance. These can be further queried directly.

Example 3.9. Using the read-resource operation at the root level

```
[standalone@localhost:9999 /]:read-resource
{
    "outcome" => "success",
    "result" => {
        "deployment" => undefined,
        "management-major-version" => 1,
        "management-minor-version" => 2,
        "name" => "hostname",
        "namespaces" => [],
        "product-name" => "EAP",
        "product-version" => "6.0.0.GA",
        "profile-name" => undefined,
        "release-codename" => "Steropes",
        "release-version" => "7.1.2.Final-redhat-1",
        "schema-locations" => [],
        "system-property" => undefined,
        "core-service" => {
            "management" => undefined,
            "service-container" => undefined,
            "server-environment" => undefined,
            "platform-mbean" => undefined
        },
        "extension" => {
            "org.jboss.as.clustering.infinispan" => undefined,
            "org.jboss.as.configadmin" => undefined,
            "org.jboss.as.connector" => undefined,
            "org.jboss.as.deployment-scanner" => undefined,
            "org.jboss.as.ee" => undefined,
            "org.jboss.as.ejb3" => undefined,
            "org.jboss.as.jaxrs" => undefined,
            "org.jboss.as.jdr" => undefined,
            "org.jboss.as.jmx" => undefined,
            "org.jboss.as.jpa" => undefined,
            "org.jboss.as.logging" => undefined,
            "org.jboss.as.mail" => undefined,
            "org.jboss.as.naming" => undefined,
            "org.jboss.as.osgi" => undefined,
            "org.jboss.as.pojo" => undefined,
            "org.jboss.as.remoting" => undefined,
            "org.jboss.as.sar" => undefined,
            "org.jboss.as.security" => undefined,
            "org.jboss.as.threads" => undefined,
            "org.jboss.as.transactions" => undefined,
            "org.jboss.as.web" => undefined,
            "org.jboss.as.webservices" => undefined,
            "org.jboss.as.weld" => undefined
        },
        "interface" => {
            "management" => undefined,
            "public" => undefined,
            "unsecure" => undefined
        },
        "path" => {
            "jboss.server.temp.dir" => undefined,
            "user.home" => undefined,
            "jboss.server.base.dir" => undefined,
            "java.home" => undefined,
            "user.dir" => undefined,
            "jboss.server.data.dir" => undefined,
            "jboss.home.dir" => undefined,
            "jboss.server.log.dir" => undefined,
            "jboss.server.config.dir" => undefined,
            "jboss.controller.temp.dir" => undefined
        },
        "socket-binding-group" => {"standard-sockets" => undefined},
        "subsystem" => {
            "logging" => undefined,
            "configadmin" => undefined,
            "datasources" => undefined,
            "deployment-scanner" => undefined,
            "ee" => undefined,
            "ejb3" => undefined,
            "infinispan" => undefined,
            "jaxrs" => undefined,
            "jca" => undefined,
            "mail" => undefined,
            "naming" => undefined,
            "osgi" => undefined,
            "remoting" => undefined,
            "threads" => undefined,
            "transactions" => undefined,
            "web" => undefined
        }
    }
}
```

```

        "jdr" => undefined,
        "jmx" => undefined,
        "jpa" => undefined,
        "mail" => undefined,
        "naming" => undefined,
        "osgi" => undefined,
        "pojo" => undefined,
        "remoting" => undefined,
        "resource-adapters" => undefined,
        "sar" => undefined,
        "security" => undefined,
        "threads" => undefined,
        "transactions" => undefined,
        "web" => undefined,
        "webservices" => undefined,
        "weld" => undefined
    }
}
}

```

2. Run the **read-resource** operation against a child node

The **read-resource** operation can be run to query child nodes from the root. The structure of the operation first defines the node to expose, and then appends the operation to run against it.

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource
```

In the following example, specific resource information about a web subsystem component can be exposed by directing the **read-resource** operation towards the specific web subsystem node.

Example 3.10. Expose child node resources from the root node

```

[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource
{
    "outcome" => "success",
    "result" => {
        "enable-lookups" => false,
        "enabled" => true,
        "executor" => undefined,
        "max-connections" => undefined,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "name" => "http",
        "protocol" => "HTTP/1.1",
        "proxy-name" => undefined,
        "proxy-port" => undefined,
        "redirect-port" => 8433,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}

```

The same results are possible by using the **cd** command to navigate into the child nodes and run the **read-resource** operation directly.

Example 3.11. Expose child node resources by changing directories

```
[standalone@localhost:9999 /] cd subsystem=web
[standalone@localhost:9999 subsystem=web] cd connector=http
[standalone@localhost:9999 connector=http] :read-resource
{
    "outcome" => "success",
    "result" => {
        "enable-lookups" => false,
        "enabled" => true,
        "executor" => undefined,
        "max-connections" => undefined,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "name" => "http",
        "protocol" => "HTTP/1.1",
        "proxy-name" => undefined,
        "proxy-port" => undefined,
        "redirect-port" => 8433,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}
```

3. Use the recursive parameter to include active values in results

The recursive parameter can be used to expose the values of all attributes, including non-persistent values, those passed at startup, or other attributes otherwise active in the runtime model.

```
[standalone@localhost:9999 /]/interface=public:read-resource(include-runtime=true)
```

Compared to the previous example, the inclusion of the **include-runtime** request property exposes additional active attributes, such as the bytes sent and bytes received by the http connector.

Example 3.12. Expose additional and active values with the `include-runtime` parameter

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-
resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "bytesReceived" => "0",
        "bytesSent" => "0",
        "enable-lookups" => false,
        "enabled" => true,
        "errorCount" => "0",
        "executor" => undefined,
        "max-connections" => undefined,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "maxTime" => "0",
        "name" => "http",
        "processingTime" => "0",
        "protocol" => "HTTP/1.1",
        "proxy-name" => undefined,
        "proxy-port" => undefined,
        "redirect-port" => 8433,
        "requestCount" => "0",
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}
```

[Report a bug](#)

3.4.7. Display Available Resource Descriptions using the Management CLI

Prerequisites

- » [Section 3.3.2, “Launch the Management CLI”](#)

Procedure 3.20. Task

1. Run the `read-resource-description` operation

From the Management CLI, use the `read-resource-description` operation to read and display the available resources. For more details on operation requests, refer to the topic [Section 3.3.7, “Use Operations and Commands in the Management CLI”](#).

```
[standalone@localhost:9999 /]:read-resource-description
```

2. Use optional parameters

The `read-resource-description` operation allows the use of the additional parameters.

- a. Use the `operations` parameter to include descriptions of the resource's operations.

```
[standalone@localhost:9999 /]:read-resource-
description(operations=true)
```

- b. Use the `inherited` parameter to include or exclude descriptions of the resource's inherited operations. The default state is true.

```
[standalone@localhost:9999 /]:read-resource-
description(inherited=false)
```

- c. Use the `recursive` parameter to include recursive descriptions of the child resources.

```
[standalone@localhost:9999 /]:read-resource-
description(recursive=true)
```

- d. Use the **locale** parameter to get the resource description in. If null, the default locale will be used.

```
[standalone@localhost:9999 /]:read-resource-description(locale=true)
```

Result

Descriptions of the available resources are displayed.

[Report a bug](#)

3.4.8. Reload the Application Server using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Procedure 3.21. Task

- ▶ **Run the reload operation**

From the Management CLI, use the **reload** operation to shut the server down via the **System.exit(0)** system call. For more details on operation requests, refer to the topic [Section 3.3.7, “Use Operations and Commands in the Management CLI”](#).

```
[standalone@localhost:9999 /]:reload
{"outcome" => "success"}
```

Result

The application server reloads by shutting down all services and starting the runtime again. The Management CLI automatically reconnects.

[Report a bug](#)

3.4.9. Shut the Application Server down using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Procedure 3.22. Task

- ▶ **Run the shutdown operation**

From the Management CLI, use the **shutdown** operation to shut the server down via the **System.exit(0)** system call. For more details on operation requests, refer to the topic [Section 3.3.7, “Use Operations and Commands in the Management CLI”](#).

```
[standalone@localhost:9999 /]:shutdown
```

Result

The application server is shut down. The Management CLI will be disconnected as the runtime is unavailable.

[Report a bug](#)

3.4.10. Configure an Attribute with the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Summary

The **write-attribute** operation is a global operation used to write or modify a selected resource attribute. You can use the operation to make persistent changes and to modify the configuration settings of your managed server instances. The request properties include the following parameters.

Request Properties

name

The name of the attribute to set the value for under the selected resource.

value

The desired value of the attribute under the selected resource. May be null if the underlying model supports null values.

Procedure 3.23. Task▶ **Run the write-attribute operation**

From the Management CLI, use the **write-attribute** operation to modify the value of a resource attribute. The operation can be run at the child node of the resource or at the root node of the Management CLI where the full resource path is specified.

Example 3.13. Disable the deployment scanner with the write-attribute operation

The following example uses the **write-attribute** operation to disable the deployment scanner. The operation is run from the root node, using tab completion to aid in populating the correct resource path.

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}
```

The results of the operation can be confirmed directly with the **read-attribute** operation.

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:read-attribute(name=scan-enabled)
{
    "outcome" => "success",
    "result" => false
}
```

The results can also be confirmed by listing all of the node's available resource attributes with the **read-resource** operation. In the following example, this particular configuration shows the **scan-enabled** attribute is now set to **false**.

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:read-resource
{
    "outcome" => "success",
    "result" => {
        "auto-deploy-exploaded" => false,
        "auto-deploy-xml" => true,
        "auto-deploy-zipped" => true,
        "deployment-timeout" => 600,
        "path" => "deployments",
        "relative-to" => "jboss.server.base.dir",
        "scan-enabled" => false,
        "scan-interval" => 5000
    }
}
```

Result

The resource attribute is updated.

[Report a bug](#)

3.5. The Management CLI Command History

3.5.1. Use the Management CLI Command History

The Management CLI features a command history functionality that is enabled by default in the

application server installation. The history is kept both as a record in the volatile memory of the active CLI session, and appended to a log file that saves automatically in the user's home directory as **.jboss-cli-history**. This history file is configured by default to record up to a maximum of 500 CLI commands.

The **history** command by itself will return the history of the current session, or with additional arguments will disable, enable or clear the history from the session memory. The Management CLI also features the ability to use your keyboard's arrow keys to go back and forth in the history of commands and operations.

Functions of the Management CLI history

- ▶ [Section 3.5.2, "Show the Management CLI Command history"](#)
- ▶ [Section 3.5.3, "Clear the Management CLI Command history"](#)
- ▶ [Section 3.5.4, "Disable the Management CLI Command history"](#)
- ▶ [Section 3.5.5, "Enable the Management CLI Command history"](#)

[Report a bug](#)

3.5.2. Show the Management CLI Command history

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)

Procedure 3.24. Task

- ▶ **Run the `history` command**

From the Management CLI, enter the **history** command:

```
[standalone@localhost:9999 /] history
```

Result

The CLI command history stored in memory since the CLI startup or the history clear command is displayed.

[Report a bug](#)

3.5.3. Clear the Management CLI Command history

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)

Procedure 3.25. Task

- ▶ **Run the `history --clear` command**

From the Management CLI, enter the **history --clear** command:

```
[standalone@localhost:9999 /] history --clear
```

Result

The history of commands recorded since the CLI startup is cleared from the session memory. The command history is still present in the **.jboss-cli-history** file saved to the user's home directory.

[Report a bug](#)

3.5.4. Disable the Management CLI Command history

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)

Procedure 3.26. Task

- ▶ **Run the `history --disable` command**

From the Management CLI, enter the **history --disable** command:

```
[standalone@localhost:9999 /] history --disable
```

Result

Commands made in the CLI will not be recorded either in memory or in the `.jboss-cli-history` file saved to the user's home directory.

[Report a bug](#)

3.5.5. Enable the Management CLI Command history

Prerequisites

- » [Section 3.3.2. "Launch the Management CLI"](#)

Procedure 3.27. Task

- » **Run the `history --enable` command**

From the Management CLI, enter the `history --enable` command:

```
[standalone@localhost:9999 /] history --enable
```

Result

Commands made in the CLI are recorded in memory and in the `.jboss-cli-history` file saved to the user's home directory.

[Report a bug](#)

Chapter 4. User Management

4.1. User Creation

4.1.1. Add the Initial User for the Management Interfaces

Overview

The management interfaces in JBoss Enterprise Application Platform 6 are secured by default, and there is no default user. This is a security precaution, to prevent security breaches from remote systems due to simple configuration errors. Local non-HTTP access is protected by a SASL mechanism, with a negotiation happening between the client and server each time the client connects for the first time from the localhost.

This task describes how to create the initial administrative user, which can use the web-based Management Console and remote instances of the Management CLI to configure and administer JBoss Enterprise Application Platform 6 from remote systems. For more information about the default security configuration, refer to [Section 9.7.1, “Default User Security Configuration”](#).



Note

HTTP communication with JBoss Enterprise Application Platform 6 is considered to be remote access, even if the traffic originates on the localhost. Therefore, you must create at least one user in order to be able to use the management console. If you attempt to access the management console before adding a user, you will receive an error because it does not even deploy until the user is added.

Procedure 4.1. Task

1. Invoke the `add-user.sh` or `add-user.bat` script.

Change to the `EAP_HOME/bin/` directory. Invoke the appropriate script for your operating system.

Red Hat Enterprise Linux

```
[user@host bin]$ ./add-user.sh
```

Microsoft Windows Server

```
C:\bin> add-user.bat
```

2. Choose to add a Management user.

Select option **a** to add a Management user. This user is added to the **ManagementRealm** and is authorized to perform management operations using the web-based Management Console or command-line based Management CLI. The other choice, **b**, adds a user to the **ApplicationRealm**, and provides no particular permissions. That realm is provided for use with applications.

3. Choose the realm for the user.

The next prompt refers to the realm where the user will be added. For a user with permissions to manage JBoss Enterprise Application Platform 6, choose the default, which is **ManagementRealm**.

4. Enter the desired username and password.

When prompted, enter the security realm, username and password. Pressing **ENTER** selects the default realm of **ManagementRealm**, which allows the user to administer JBoss Enterprise Application Platform 6 using the management interfaces. You must add at least one user to this realm. You are prompted to confirm the information. If you are satisfied, type **yes**.

5. Choose whether the user represents a remote JBoss Enterprise Application Platform 6 server instance.

Besides administrators, the other type of user which occasionally needs to be added to JBoss Enterprise Application Platform 6 in the **ManagementRealm** is a user representing another instance of JBoss Enterprise Application Platform 6, which needs to be able to authenticate to join a cluster as a member. The next prompt allows you to designate your added user for this

purpose. If you select **yes**, you will be given a hashed **secret** value, representing the user's password, which would need to be added to a different configuration file. For the purposes of this task, answer **no** to this question.

6. Enter additional users.

You can enter additional users if desired, by repeating the procedure. You can also add them at any time on a running system. Instead of choosing the default security realm, you can add users to other realms to fine-tune their authorizations.

7. Create users non-interactively.

You can create users non-interactively, by passing in each parameter at the command line. This approach is not recommended on shared systems, because the passwords will be visible in log and history files. The syntax for the command, using the management realm, is:

```
[user@host bin]$ ./add-user.sh username password
```

To use the application realm, use the **-a** parameter.

```
[user@host bin]$ ./add-user.sh -a username password
```

Result

Any users you add are activated within the security realms you have specified. Users active within the **ManagementRealm** realm are able to manage JBoss Enterprise Application Platform 6 from remote systems.

[Report a bug](#)

4.1.2. Add a User to the Management Interface

Use the same procedure outlined in [Section 4.1.1, “Add the Initial User for the Management Interfaces”](#).

[Report a bug](#)

Chapter 5. Network and Port Configuration

5.1. Interfaces

5.1.1. About Interfaces

The application server uses named interface references throughout the configuration. This gives the configuration the ability to reference the individual interface declarations with logical names, rather than the full details of the interface at each use. The use of logical names also allows for consistency in group references to named interfaces, where server instances on a managed domain may contain varying interface details across multiple machines. With this methodology, each server instance may correspond to a logical name group that allows for easier administration of the interface group as a whole.

A network interface is declared by specifying a logical name and a selection criteria for the physical interface. The application server ships with a default configuration for a management and a public interface name. In this configuration, the public interface group is intended for use by any application-related network communication such as Web or Messaging. The management interface group is intended for use for all components and services that are required by the management layer, including the HTTP Management Endpoint. The interface names themselves are provided as a suggestion only, where any name for any group can be substituted or created as required.

The **domain.xml**, **host.xml** and **standalone.xml** configuration files all include interface declarations. The declaration 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. The following examples show multiple possible configurations of interface declarations, typically defined in either the **standalone.xml** or **host.xml** configuration files. This allows any remote host groups to maintain their own specific interface attributes, while still allowing reference to the any interface groups in the **domain.xml** configuration file of the domain controller.

The first example shows a specific **inet-address** value specified for both the **management** and **public** relative name groups.

Example 5.1. An interface group created with an inet-address value

```
<interfaces>
  <interface name="management">
    <inet-address value="127.0.0.1"/>
  </interface>
  <interface name="public">
    <inet-address value="127.0.0.1"/>
  </interface>
</interfaces>
```

In the following example a global interface group uses the **any-address** element to declare a wildcard address.

Example 5.2. A global group created with a wildcard declaration

```
<interface name="global">
  <!-- Use the wild-card address -->
  <any-address/>
</interface>
```

The following example declares a network interface card under a relative group with the name **external**.

Example 5.3. An external group created with an NIC value

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

In the following example a declaration is created as the default group for a specific requirement. In this instance, the characteristics of the additional elements set the condition for the interface to be a valid match. This allows for the creation of very specific interface declaration groups, with the ability to reference them in a preset manner, reducing the configuration and administration time across multiple server instances.

Example 5.4. A default group created with specific conditional values

```
<interface name="default">
    <!-- Match any interface/address on the right subnet if it's
        up, supports multicast, and isn't point-to-point -->
    <subnet-match value="192.168.0.0/16"/>
    <up/>
    <multicast/>
    <not>
        <point-to-point/>
    </not>
</interface>
```

While the interface declarations can be made and edited in the source configuration files, the Management CLI and Management Console provide a safe, controlled and persistent environment for configuration changes.

[Report a bug](#)

5.1.2. Configure Interfaces

The default interface configurations in the `standalone.xml` and `host.xml` configuration files offer three named interfaces with relative interface tokens for each. You can use the Management Console or Management CLI to configure additional attributes and values, as listed in the table below. You can also replace the relative interface bindings with specific values as required. Note that if you do so, you will be unable to pass an interface value at server runtime, as the `-b` switch can only override a relative value.

Example 5.5. 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="unsecure">
        <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
    </interface>
</interfaces>
```

Table 5.1. Interface Attributes and Values

Interface Element	Description
any	Empty element of the address exclusion type, used to constrain the selection 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 <code>java.net.preferIPv4Stack</code> 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.
any-ipv4-address	Empty element indicating that sockets using this interface should be bound to the IPv4 wildcard address (0.0.0.0).
any-ipv6-address	Empty element indicating that sockets using this interface should be bound to the IPv6 wildcard address (:).
inet-address	Either a IP address in IPv6 or IPv4 dotted decimal notation, or a hostname 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 <code>inet-addressType</code> 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.
nic	The name of a network interface (e.g. <code>eth0</code> , <code>eth1</code> , <code>lo</code>).
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	Empty element of the address exclusion type, used to constrain the selection 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.
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 "slash notation"; e.g. "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.

Configure Interface Attributes

Choose either the Management CLI or the Management Console to configure your interface attributes as required.

A. Configure Interface Attributes with the Management CLI

Use the Management CLI to add new interfaces and write new values to the interface attributes.

1. Add a New Interface

Use the `add` operation to create a new interface if required. You can run this command from the root of the Management CLI session, which in the following example creates a new interface name title `interfacename`, with an `inet-address` declared as `12.0.0.2`.

```
/interface=interfacename/:add(inet-address=12.0.0.2)
```

2. Edit Interface Attributes

Use the **write** operation to write a new value to an attribute. You can use tab completion to help complete the command string as you type, as well as to expose the available attributes. The following example updates the **inet-address** value to **12.0.0.8**

```
/interface=interfacename/:write(inet-address=12.0.0.8)
```

3. Edit Interface Attributes

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[standalone@localhost:9999 interface=public] :read-resource(include-runtime=true)
```

B. Configure Interface Attributes with the Management Console

Use the Management Console to add new interfaces and write new values to the interface attributes.

1. Log into the Management Console.

Log into the Management Console of your Managed Domain or Standalone Server instance.

2. If you use a Managed Domain, choose the correct profile.

Select the **Profiles** tab at the top right, and then select the correct profile from the **Profile** menu at the top left of the next screen.

3. Select the Interfaces item from the navigation menu.

Select the **Interfaces** menu item from the navigation menu.

4. Add a New Interface

- a. Click the **Add** button.
- b. Enter any required values for **Name**, **Inet Address** and **Address Wildcard**.
- c. Click the **Save** to finish.

5. Edit Interface Attributes

- a. Select the Interface to edit and click the **Edit** button.
- b. Enter any required values for **Name**, **Inet Address** and **Address Wildcard**.
- c. Click the **Save** to finish.

[Report a bug](#)

5.2. Socket Binding Groups

5.2.1. About Socket Binding Groups

Socket bindings and socket binding groups allow you to define network ports and their relationship to the networking interfaces required for your JBoss Enterprise Application Platform 6 configuration.

A socket binding is a named configuration for a socket. The declarations for these named configurations can be found in both the **domain.xml** and **standalone.xml** configuration files. Other sections of the configuration can then reference those sockets by their logical name, rather than having to include the full details of the socket configuration. This allows you to reference relative socket configurations which may otherwise vary on different machines.

Socket bindings are collected under a socket binding group. A socket binding group is a collection of socket binding declarations that are grouped under a logical name. The named group can then be referenced throughout the configuration. A standalone server contains only one such group, while a managed domain instance 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 naming groups allow for simplified references to be used for particular groups of socket bindings when configuring server groups in the case of a managed domain. Another common use is for the configuration and management of multiple instances of the standalone server on the one system. The following examples show the default socket binding groups in the configuration files for the standalone and domain instances.

Example 5.6. Default socket bindings for the standalone configuration

The default socket binding groups in the **standalone.xml** configuration file are grouped under **standard-sockets**. This group is also referenced to the **public** interface, using the same logical referencing methodology.

```
<socket-binding-group name="standard-sockets" default-interface="public">
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jacorb" port="3528"/>
    <socket-binding name="jacorb-ssl" port="3529"/>
    <socket-binding name="jmx-connector-registry" port="1090"
interface="management"/>
    <socket-binding name="jmx-connector-server" port="1091"
interface="management"/>
    <socket-binding name="jndi" port="1099"/>
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-throughput" port="5455"/>
    <socket-binding name="osgi-http" port="8090" interface="management"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
</socket-binding-group>
```

Example 5.7. Default socket bindings for the domain configuration

The default socket binding groups in the `domain.xml` configuration file contain four groups: the **standard-sockets**, **ha-sockets**, **full-sockets** and the **full-ha-sockets** groups. These groups are also referenced to an interface called **public**.

```

<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="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="osgi-http" interface="management" port="8090"/>
        <socket-binding name="remoting" port="4447"/>
        <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 name="ajp" port="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="jgroups-mping" port="0" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
        <socket-binding name="jgroups-tcp" port="7600"/>
        <socket-binding name="jgroups-tcp-fd" port="57600"/>
        <socket-binding name="jgroups-udp" port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
        <socket-binding name="jgroups-udp-fd" port="54200"/>
        <socket-binding name="modcluster" port="0" multicast-
address="224.0.1.105" multicast-port="23364"/>
        <socket-binding name="osgi-http" interface="management" port="8090"/>
        <socket-binding name="remoting" port="4447"/>
        <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="full-sockets" default-interface="public">
        <!-- Needed for server groups using the 'full' profile -->
        <socket-binding name="ajp" port="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="jacorb" interface="unsecure" port="3528"/>
        <socket-binding name="jacorb-ssl" interface="unsecure" port="3529"/>
        <socket-binding name="messaging" port="5445"/>
        <socket-binding name="messaging-group" port="0" multicast-
address="${jboss.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
        <socket-binding name="messaging-throughput" port="5455"/>
        <socket-binding name="osgi-http" interface="management" port="8090"/>
        <socket-binding name="remoting" port="4447"/>
        <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="full-ha-sockets" default-interface="public">
        <!-- Needed for server groups using the 'full-ha' profile -->
        <socket-binding name="ajp" port="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="jacorb" interface="unsecure" port="3528"/>
        <socket-binding name="jacorb-ssl" interface="unsecure" port="3529"/>
        <socket-binding name="jgroups-mping" port="0" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
        <socket-binding name="jgroups-tcp" port="7600"/>
        <socket-binding name="jgroups-tcp-fd" port="57600"/>
        <socket-binding name="jgroups-udp" port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
        <socket-binding name="jgroups-udp-fd" port="54200"/>
        <socket-binding name="messaging" port="5445"/>
        <socket-binding name="messaging-group" port="0" multicast-
address="${jboss.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
        <socket-binding name="messaging-throughput" port="5455"/>
    </socket-binding-group>

```

```

<socket-binding name="modcluster" port="0" multicast-
address="224.0.1.105" multicast-port="23364"/>
<socket-binding name="osgi-http" interface="management" port="8090"/>
<socket-binding name="remoting" port="4447"/>
<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-groups>

```

The socket binding instances can be created and edited in the **standalone.xml** and **domain.xml** source files in the application server directory. The recommended method of managing bindings is to use either the Management Console or the Management CLI. The advantages of using the Management Console include a graphical user interface with a dedicated Socket Binding Group screen under the **General Configuration** section. The Management CLI offers an API and workflow based around a command line approach that allows for batch processing and the use of scripts across the higher and lower levels of the application server configuration. Both interfaces allow for changes to be persisted or otherwise saved to the server configuration.

[Report a bug](#)

5.2.2. Configure Socket Bindings

Socket bindings can be defined in unique socket binding groups. The Standalone Server contains one such group, the **standard-sockets** group, and is unable to create any further groups. Instead you can create alternate Standalone Server configuration files. For the Managed Domain however, you can create multiple socket binding groups and configure the socket bindings that they contain as you require. The following table shows the available attributes for each socket binding.

Table 5.2. Socket Binding Attributes

Component	Description	Role
Name	Logical name of the socket configuration that should be used elsewhere in the configuration.	Required
Port	Base port to which a socket based on this configuration should be bound. Note that servers can be configured to override this base value by applying an increment or decrement to all port values.	Required
Interface	Logical name of the interface to which a socket based on this configuration should be bound. If not defined, the value of the "default-interface" attribute from the enclosing socket binding group will be used.	Optional
Multicast Address	If the socket will be used for multicast, the multicast address to use.	Optional
Multicast Port	Bound to the lifecycle of the conversation. The conversation scope is between the lengths of the request and the session, and is controlled by the application.	Optional
Fixed Port	If the above contexts do not meet your needs, you can define custom scopes.	Optional

▶ **Configure Socket Bindings in Socket Binding Groups**

Choose either the Management CLI or the Management Console to configure your socket bindings as required.

A. Configure Socket Bindings Using the Management CLI

Use the Management CLI to configure socket bindings.

1. Add a New Socket Binding

Use the **add** operation to create a new address setting if required. You can run this command from the root of the Management CLI session, which in the following examples creates a new socket binding titled **newsocket**, with a **port** attribute declared as **1234**. The examples apply for both Standalone Server and a Managed Domain editing on the **standard-sockets** socket binding group as shown.

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:add(port=1234)
```

2. Edit Pattern Attributes

Use the **write** operation to write a new value to an attribute. You can use tab completion to help complete the command string as you type, as well as to expose the available attributes. The following example updates the **port** value to **2020**

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:write-attribute(name=port,value=2020)
```

3. Confirm Pattern Attributes

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:read-resource
```

B. Configure Socket Bindings Using the Management Console

Use the Management Console to configure socket bindings.

1. Log into the Management Console.

Log into the Management Console of your Managed Domain or Standalone Server.

2. Select the Profile tab

Select the **Profiles** tab at the top right.

3. Select the Socket Binding item from the navigation menu.

Select the **Socket Binding** menu item from the navigation menu. If you are using a Managed Domain, select the desired group in the **Socket Binding Groups** menu.

4. Add a New Socket Binding

- a. Click the **Add** button.
- b. Enter any required values for **Name**, **Port** and **Binding Group**.
- c. Click the **Save** to finish.

5. Edit Interface Attributes

- a. Select the Socket Binding to edit and click the **Edit** button.
- b. Enter any required values such as **Name**, **Interface** or **Port**.
- c. Click the **Save** to finish.

[Report a bug](#)

5.2.3. Network Ports Used By JBoss Enterprise Application Platform 6

The ports used by the JBoss Enterprise Application Platform 6 default configuration depend on several factors:

- ▶ Whether you use a Managed Domain or Standalone Server configuration.
- ▶ Whether your server groups use one of the default socket binding groups, or a custom group.
- ▶ The requirements of your individual deployments.



Numerical port offsets

A numerical port offset can be configured, to alleviate port conflicts when you run multiple servers on the same physical server. If your server uses a numerical port offset, add the offset to the default port number for its server group's socket binding group. For instance, if the HTTP port of the socket binding group is 8080, and your server uses a port offset of 100, its HTTP port is 8180.

Unless otherwise stated, the ports use the TCP protocol.

The default socket binding groups

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

Table 5.3. Reference of the default socket bindings

Name	Port	Multicast Port	Description	full-ha-sockets	full-sockets	ha-socket	standard-socket
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.	Yes	Yes	Yes	Yes
http	8080		The default port for deployed web applications.	Yes	Yes	Yes	Yes
https	8443		SSL-encrypted connection between deployed web applications and clients.	Yes	Yes	Yes	Yes
jacorb	3528		CORBA services for JTS transactions and other ORB-dependent services.	Yes	Yes	No	No
jacorb-ssl	3529		SSL-encrypted CORBA services.	Yes	Yes	No	No
jgroup-s-diagonistics	7500		Multicast. Used for peer discovery in HA clusters.	Yes	No	Yes	No
jgroup-s-mping	45700		Multicast. Used to discover initial membership in a HA cluster.	Yes	No	Yes	No
jgroups-tcp	7600		Unicast peer discovery in HA clusters using TCP.	Yes	No	Yes	No
jgroups-tcp-fd	57600		Used for HA failure detection over TCP.	Yes	No	Yes	No
jgroups-udp	55200	45688	Unicast peer discovery in HA clusters using UDP.	Yes	No	Yes	No
jgroups-udp-fd	54200		Used for HA failure detection over UDP.	Yes	No	Yes	No
messaging	5445		JMS service.	Yes	Yes	No	No
messaging-group			Referenced by HornetQ JMS broadcast and discovery groups.	Yes	Yes	No	No
messaging-throughput	5455		Used by JMS Remoting.	Yes	Yes	No	No
mod_cluster	23364		Multicast port for communication between the JBoss Enterprise Application Platform and the HTTP load balancer.	Yes	No	Yes	No
osgi-http	8090		Used by internal components which use the OSGi	Yes	Yes	Yes	Yes

		subsystem.			
remote ng	4447	Used for remote EJB invocation.	Yes	Yes	Yes
txnr- recovery- enviro- nment	4712	The JTA transaction recovery manager.	Yes	Yes	Yes
txn- status- manager	4713	The JTA / JTS transaction manager.	Yes	Yes	Yes

Management Ports

In addition to the socket binding groups, each host controller opens two more ports for management purposes:

- ▶ 9990 - The Web Management Console port
- ▶ 9999 - The port used by the Management Console and Management API

[Report a bug](#)

5.2.4. About Port Offsets for Socket Binding Groups

Port offsets are a numeric offset added to the port values given by the socket binding group for that server. This allows a single server to inherit the socket bindings of the server group that it belongs, with an offset to ensure that it does not clash with the other servers in the group. For instance, if the HTTP port of the socket binding group is 8080, and your server uses a port offset of 100, its HTTP port is 8180.

[Report a bug](#)

5.2.5. Configure Port Offsets

Configure Port Offsets

Choose either the Management CLI or the Management Console to configure your port offsets.

A. Configure Port Offsets Using the Management CLI

Use the Management CLI to configure port offsets.

1. Edit Port Offsets

Use the **write** operation to write a new value to the port offset attribute. The following example updates the **socket-binding-port-offset** value of **server-two** to **250**. This server is a member of the default local host group.

```
[domain@localhost:9999 /] /host=master/server-config=server-two/:write-attribute(name=socket-binding-port-offset,value=250)
```

2. Confirm Port Offset Attributes

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[domain@localhost:9999 /] /host=master/server-config=server-two/:read-resource(include-runtime=true)
```

B. Configure Port Offsets Using the Management Console

Use the Management Console to configure port offsets.

1. Log into the Management Console.

Log into the Management Console of your Managed Domain.

2. Select the Server tab

Select the **Server** tab at the top right.

3. Edit Port Offset Attributes

- a. Select the server under the **Configuration Name** section and click the **Edit** button.
- b. Enter any desired values in the **Port Offset** field.
- c. Click the **Save** button to finish.

[Report a bug](#)

5.3. IPv6

5.3.1. Configure JVM Stack Preferences for IPv6 Networking

Task Summary

This topic covers enabling IPv6 networking for the JBoss Enterprise Application Platform 6 installation.

Procedure 5.1. Task

1. Open the relevant file for the installation:
 - A. **For a Standalone Server:**
Open `EAP_HOME/bin/standalone.conf`.
 - B. **For a Managed Domain:**
Open `EAP_HOME/bin/domain.conf`.
2. Change the IPv4 Stack Java property to false:

`-Djava.net.preferIPv4Stack=false`

For example:

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
Djava.net.preferIPv4Stack=false
  -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000
  -Dsun.rmi.dgc.server.gcInterval=3600000 -
Djava.net.preferIPv6Addresses=true"
fi
```

[Report a bug](#)

5.3.2. Configure the Interface Declarations for IPv6 Networking

Task Summary

Follow these steps to configure the interface inet address to the IPv6 default:

Prerequisites

- » [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#)
- » [Section 3.2.2, “Log in to the Management Console”](#)

Procedure 5.2. Task

1. Select the **Profile** tab, in the top right corner of the console.
2. Select the **Interfaces** option under **General Configuration**.
3. Select the named network interface to configure.
4. Click the **Edit** button.
5. Set the inet address to:

`${jboss.bind.address.management:[ADDRESS]}`

6. Click the **Save** button to save the changes.
7. Restart the server to implement the changes.

[Report a bug](#)

5.3.3. Configure JVM Stack Preferences for IPv6 Addresses

Task Summary

This topic covers configuring the JBoss Enterprise Application Platform 6 installation to prefer IPv6 addresses through the configuration files.

Procedure 5.3. Task

1. Open the relevant file for the installation:

A. For a Standalone Server:

Open *EAP_HOME/bin/standalone.conf*.

B. For a Managed Domain:

Open *EAP_HOME/bin/domain.conf*.

2. Append the following Java property to the Java VM options:

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

For example:

```
# Specify options to pass to the Java VM.  
#  
if [ "$JAVA_OPTS" = "x" ]; then  
    JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -  
    Djava.net.preferIPv4Stack=false  
    -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000  
    -Dsun.rmi.dgc.server.gcInterval=3600000 -  
    Djava.net.preferIPv6Addresses=true"  
fi
```

[Report a bug](#)

Chapter 6. Datasource Management

6.1. Introduction

6.1.1. 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.

The JDBC 4.0 specification is defined here: <http://jcp.org/en/jsr/detail?id=221>.

To get started with JDBC and datasources, refer to the JDBC Driver section of the Administration and Configuration Guide for JBoss Enterprise Application Platform 6.

[Report a bug](#)

6.1.2. JBoss Enterprise Application Platform 6 Supported Databases

The list of JDBC compliant databases supported by JBoss Enterprise Application Platform 6 is available here: <http://www.redhat.com/resource/library/articles/jboss-enterprise-application-platform-supported-configurations>.

[Report a bug](#)

6.1.3. Types of Datasources

The two general types of resources are referred to as **datasources** and **XA datasources**.

Non-XA datasources are used for applications which do not use transactions, or applications which use transactions with a single database.

XA datasources are used by applications whose transactions are distributed across multiple databases. XA datasources introduce additional overhead.

You specify the type of your datasource when you create it in the Management Console or Management CLI.

[Report a bug](#)

6.1.4. The Example Datasource

JBoss Enterprise Application Platform 6 includes a H2 database. It is a lightweight, relational database management system that provides developers with the ability to quickly build applications, and is the example datasource for the platform.



Warning

However, it should not be used in a production environment. It 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.

For a list of supported and certified datasources, refer here: [Section 6.1.2, “JBoss Enterprise Application Platform 6 Supported Databases”](#).

[Report a bug](#)

6.1.5. Deployment of -ds.xml files

In JBoss Enterprise Application Platform 6, datasources are defined as a resource of the server subsystem. In previous versions, a ***-ds.xml** datasource configuration file was required in the deployment directory of the server configuration. ***-ds.xml** files can still be deployed in JBoss Enterprise Application Platform 6, following the schema available here:

http://docs.jboss.org/ironjacamar/schema/datasources_1_1.xsd.

**Warning**

This feature should only be used for development. It is not recommended for production environments, because it is not supported by the JBoss administrative and management tools.

**Important**

It is mandatory to use a reference to an already deployed / defined <driver> entry when deploying * -ds.xml files.

[Report a bug](#)

6.2. JDBC Drivers

6.2.1. Install a JDBC Driver with the Management Console

Summary

Before your application can connect to a JDBC datasource, your datasource vendor's JDBC drivers need to be installed in a location where the JBoss Enterprise Application Platform can use them. JBoss Enterprise Application Server allows you to deploy these drivers just like any other deployment. This means that you can deploy them across multiple servers in a server group, if you use a managed domain.

**Note**

The preferred installation method for JDBC drivers is to install them as a core module. To install the JDBC driver as a core module, refer here: [Section 6.2.2, “Install a JDBC Driver as a Core Module”](#).

Prerequisites

Before performing this task, you need to meet the following prerequisites:

- » Download the JDBC driver from your database vendor.

Procedure 6.1. Task

1. **Access the Management Console.**

[Section 3.2.2, “Log in to the Management Console”](#)

2. **Deploy the JAR file to your server or server group.**

If you use a managed domain, deploy the JAR file to a server group. Otherwise, deploy it to your server. See [Section 8.2.2, “Deploy an Application Using the Management Console”](#).

Result:

The JDBC driver is deployed, and is available for your applications to use.

[Report a bug](#)

6.2.2. Install a JDBC Driver as a Core Module

Prerequisites

Before performing this task, you need to meet the following prerequisites:

- » Download the JDBC driver from your database vendor. JDBC driver download locations are listed here: [Section 6.2.3, “JDBC Driver Download Locations”](#).
- » Extract the archive.

Procedure 6.2. Task

1. Create a file path structure under the **EAP_HOME/modules/** directory. For example, for a MySQL JDBC driver, create a directory structure as follows: **EAP_HOME/modules/com/mysql/main/**.
2. Copy the JDBC driver JAR into the **main/** subdirectory.
3. In the **main/** subdirectory, create a **module.xml** file similar to the example below:

Example 6.1. Example module.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.15.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

The module name, **com.mysql**, should match the directory structure for the module.

4. Start the Server.
5. Start the Management CLI.
6. Run the following CLI command to add the JDBC driver module as a driver:

```
/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)
```

Example 6.2. Example CLI Command

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)
```

Result

The JDBC driver is now installed and set up as a core module, and is available to be referenced by application datasources.

[Report a bug](#)

6.2.3. JDBC Driver Download Locations

The following table gives the standard download locations for JDBC drivers of common databases used with the JBoss Enterprise Application Platform. 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 6.1. JDBC driver download locations

Vendor	Download Location
MySQL	http://www.mysql.com/products/connector/
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
IBM	http://www-306.ibm.com/software/data/db2/java/
Sybase	http://www.sybase.com/products/allproducts/ads/softwaredeveloperkit/jconnect
Microsoft	http://msdn.microsoft.com/data/jdbc/

[Report a bug](#)

6.2.4. Access Vendor Specific Classes

Summary

This topic covers the steps required to use the JDBC specific classes. This is necessary when an application needs to use vendor specific functionality that is not part of the JDBC API.



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.



Important

Follow the vendor specific API guidelines closely, as the connection is being controlled by the IronJacamar container.

Prerequisites

- » [Section 6.2.2, “Install a JDBC Driver as a Core Module”.](#)

Procedure 6.3. Add a Dependency to the Application

A. Configure the MANIFEST.MF file

1. Open the application's **META-INF/MANIFEST.MF** file in a text editor.
2. Add a dependency for the JDBC module and save the file.

Dependencies: **MODULE_NAME**

Example 6.3. Example Dependency

Dependencies: com.mysql

B. 1. Create a jboss-deployment-structure.xml file

Create a file called **jboss-deployment-structure.xml** in the **META-INF/** or **WEB-INF** folder of the application.

Example 6.4. Example jboss-deployment-structure.xml file

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Example 6.5. Access the Vendor Specific API

The example below accesses the MySQL API.

```
import java.sql.Connection;
import org.jboss.jca.adapters.jdbc.WrappedConnection;

Connection c = ds.getConnection();
WrappedConnection wc = (WrappedConnection)c;
com.mysql.jdbc.Connection mc = wc.getUnderlyingConnection();
```

[Report a bug](#)

6.3. Non-XA Datasources

6.3.1. Create a Non-XA Datasource with the Management Interfaces

Task Summary

This topic covers the steps required to create a non-XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- ▶ The JBoss Enterprise Application Platform 6 server must be running.

Oracle Datasources

Prior to version 10.2 of the Oracle datasource, the <no-tx-separate-pools/> parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.

Procedure 6.4. Task

▶ A. Management CLI

1. Launch the CLI tool and connect to your server.
2. Run the following command to create a non-XA datasource, configuring the variables as appropriate:

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --connection-url=CONNECTION_URL
```

3. Enable the datasource:

```
data-source enable --name=DATASOURCE_NAME
```

B. Management Console

1. Login to the Management Console.

2. **Navigate to the Datasources panel in the Management Console**

- a. **A. Standalone Mode**

Select the **Profile** tab from the top-right of the console.

- B. Domain Mode**

- a. Select the **Profiles** tab from the top-right of the console.

- b. Select the appropriate profile from the drop-down box in the top left.

- c. Expand the **Subsystems** menu on the left of the console.

- b. Select **Connector** → **Datasources** from the menu on the left of the console.

Figure 6.1. Datasources panel

3. Create a new datasource

- Select the **Add** button at the top of the **Datasources** panel.
- Enter the new datasource attributes in the **Create Datasource** wizard and proceed with the **Next** button.
- Enter the JDBC driver details in the **Create Datasource** wizard and proceed with the **Next** button.
- Enter the connection settings in the **Create Datasource** wizard and select the **Done** button.

Result

The non-XA datasource has been added to the server. It is now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

[Report a bug](#)

6.3.2. Modify a Non-XA Datasource with the Management Interfaces

Task Summary

This topic covers the steps required to modify a non-XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- ▶ [Section 2.6.1 “Start JBoss Enterprise Application Platform 6”.](#)



JTA Integration

Non-XA datasources can be integrated with JTA transactions. To integrate the datasource with JTA, ensure that the **jta** parameter is set to **true**.

Procedure 6.5. Task

- ▶ A. **Management CLI**
 1. [Section 3.3.2, “Launch the Management CLI”](#)
 2. Use the **write-attribute** command to configure a datasource attribute:

```
/subsystem=datasources/data-source=DATASOURCE_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

3. Reload the server to confirm the changes:

```
:reload
```

B. Management Console

1. [Section 3.2.2, “Log in to the Management Console”.](#)
2. [Navigate to the Datasources panel in the Management Console](#)
 - a. A. Standalone Mode
Select the **Profile** tab from the top-right of the console.
 - b. Domain Mode
 - a. Select the **Profiles** tab from the top-right of the console.
 - b. Select the appropriate profile from the drop-down box in the top left.
 - c. Expand the **Subsystems** menu on the left of the console.
 - b. Select **Connector** → **Datasources** from the menu on the left of the console.

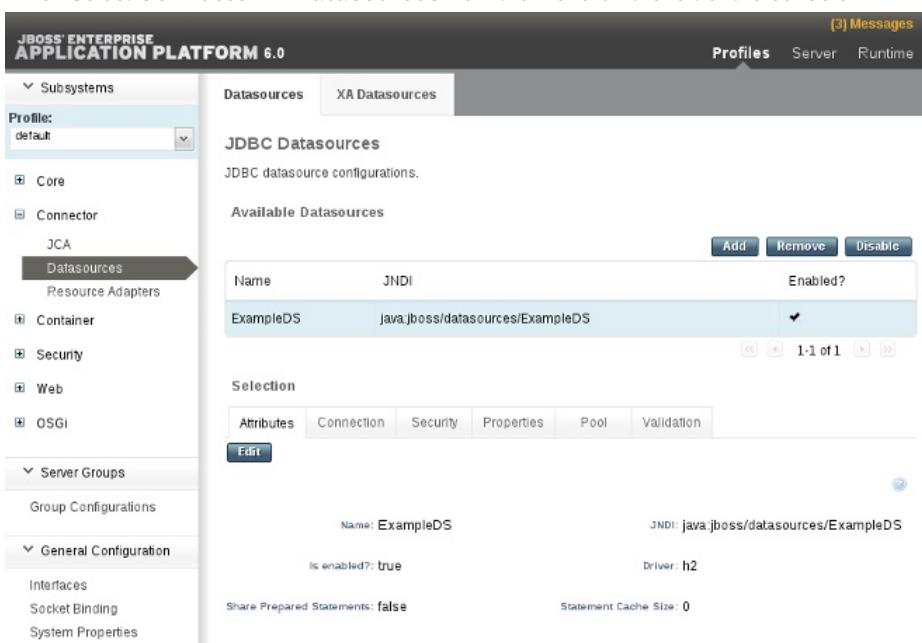


Figure 6.2. Datasources panel

3. Edit the datasource

- a. Select the relevant datasource from the **Available Datasources** list. The datasource attributes are displayed in the **Attributes** panel below it.
- b. Select the **Edit** button to edit the datasource attributes.
- c. Edit the datasource attributes and select the **Save** button when done.

Result

The non-XA datasource has been configured. The changes are now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

- ▶ To create a new datasource, refer here: [Section 6.3.1, “Create a Non-XA Datasource with the Management Interfaces”.](#)
- ▶ To remove the datasource, refer here: [Section 6.3.3, “Remove a Non-XA Datasource with the Management Interfaces”.](#)

[Report a bug](#)

6.3.3. Remove a Non-XA Datasource with the Management Interfaces

Task Summary

This topic covers the steps required to remove a non-XA datasource from JBoss Enterprise Application Platform 6, using either the Management Console or the Management CLI.

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”.](#)

Procedure 6.6. Task

▶ A. Management CLI

1. [Section 3.3.2, “Launch the Management CLI”.](#)
2. Run the following command to remove a non-XA datasource:

```
data-source remove --name=DATASOURCE_NAME
```

B. Management Console

1. [Section 3.2.2, “Log in to the Management Console”.](#)

2. **Navigate to the Datasources panel in the Management Console**

- a. **A. Standalone Mode**

Select the **Profile** tab from the top-right of the console.

- b. **Domain Mode**

- a. Select the **Profiles** tab from the top-right of the console.
- b. Select the appropriate profile from the drop-down box in the top left.
- c. Expand the **Subsystems** menu on the left of the console.

- b. Select **Connector → Datasources** from the menu on the left of the console.

Name	JNDI	Enabled?
ExampleDS	java:jboss/datasources/ExampleDS	<input checked="" type="checkbox"/>

Selection

Attributes	Connection	Security	Properties	Pool	Validation
<input type="button" value="Edit"/> Name: ExampleDS JNDI: java:jboss/datasources/ExampleDS Is enabled?: true Driver: h2 Share Prepared Statements: false Statement Cache Size: 0					

Figure 6.3. Datasources panel

3. Select the registered datasource to be deleted, and click the **Remove** button in the top right corner of the console.

Result

The non-XA datasource has been removed from the server.

- ▶ To add a new datasource, refer here: [Section 6.3.1, “Create a Non-XA Datasource with the Management Interfaces”.](#)

[Report a bug](#)

6.4. XA Datasources

6.4.1. Create an XA Datasource with the Management Interfaces

Task Summary

This topic covers the steps required to create an XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- » [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”.](#)



Oracle Datasources

Prior to version 10.2 of the Oracle datasource, the <no-tx-separate-pools/> parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.

Procedure 6.7. Task

A. Management CLI

1. [Section 3.3.2, “Launch the Management CLI”.](#)
2. Run the following command to create an XA datasource, configuring the variables as appropriate:

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-name=JNDI_NAME --
driver-name=DRIVER_NAME --xa-datasource-class=XA_DATASOURCE_CLASS
```

3. Configure the XA datasource properties

a. Set the server name

Run the following command to configure the server name for the host:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-
datasource-properties=ServerName:add(value=HOSTNAME)
```

b. Set the database name

Run the following command to configure the database name:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-
datasource-properties=DatabaseName:add(value=DATABASE_NAME)
```

4. Enable the datasource:

```
xa-data-source enable --name=XA_DATASOURCE_NAME
```

B. Management Console

1. [Section 3.2.2, “Log in to the Management Console”.](#)

2. Navigate to the Datasources panel in the Management Console

a. A. Standalone Mode

Select the **Profile** tab from the top-right of the console.

B. Domain Mode

- a. Select the **Profiles** tab from the top-right of the console.
- b. Select the appropriate profile from the drop-down box in the top left.
- c. Expand the **Subsystems** menu on the left of the console.

- b. Select **Connector** → **Datasources** from the menu on the left of the console.

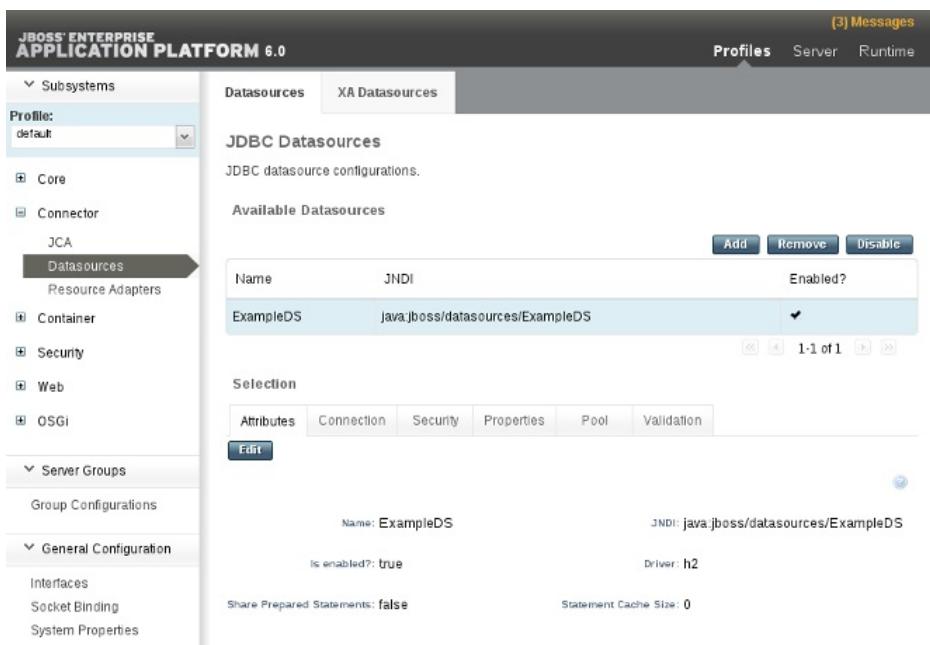


Figure 6.4. Datasources panel

3. Select the **XA Datasource** panel.
4. **Create a new XA datasource**
 - a. Select the **Add** button at the top of the **XA Datasources** panel.
 - b. Enter the new XA datasource attributes in the **Create XA Datasource** wizard and proceed with the **Next** button.
 - c. Enter the JDBC driver details in the **Create XA Datasource** wizard and proceed with the **Next** button.
 - d. Edit the XA properties and proceed with the **Next** button.
 - e. Enter the connection settings in the **Create XA Datasource** wizard and select the **Done** button.

Result

The XA datasource has been added to the server. It is now visible in either the `standalone.xml` or `domain.xml` file, as well as the management interfaces.

- ▶ To configure the datasource further, refer here: [Section 6.4.2, “Modify an XA Datasource with the Management Interfaces”](#).
- ▶ To remove the datasource, refer here: [Section 6.4.3, “Remove an XA Datasource with the Management Interfaces”](#).

[Report a bug](#)

6.4.2. Modify an XA Datasource with the Management Interfaces

Task Summary

This topic covers the steps required to modify an XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#).

Procedure 6.8. Task

- ▶ A. **Management CLI**

1. [Section 3.3.2, “Launch the Management CLI”](#).

2. **Configure XA datasource attributes**

Use the `write-attribute` command to configure a datasource attribute:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

3. Configure XA datasource properties

Run the following command to configure an XA datasource subresource:

```
/subsystem=datasources/xa-data-source=DATASOURCE_NAME/xa-datasource-
properties=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

4. Reload the server to confirm the changes:

```
:reload
```

B. Management Console

1. [Section 3.2.2, “Log in to the Management Console”.](#)

2. Navigate to the Datasources panel in the Management Console

a. A. Standalone Mode

Select the **Profile** tab from the top-right of the console.

B. Domain Mode

a. Select the **Profiles** tab from the top-right of the console.

b. Select the appropriate profile from the drop-down box in the top left.

c. Expand the **Subsystems** menu on the left of the console.

b. Select **Connector** → **Datasources** from the menu on the left of the console.

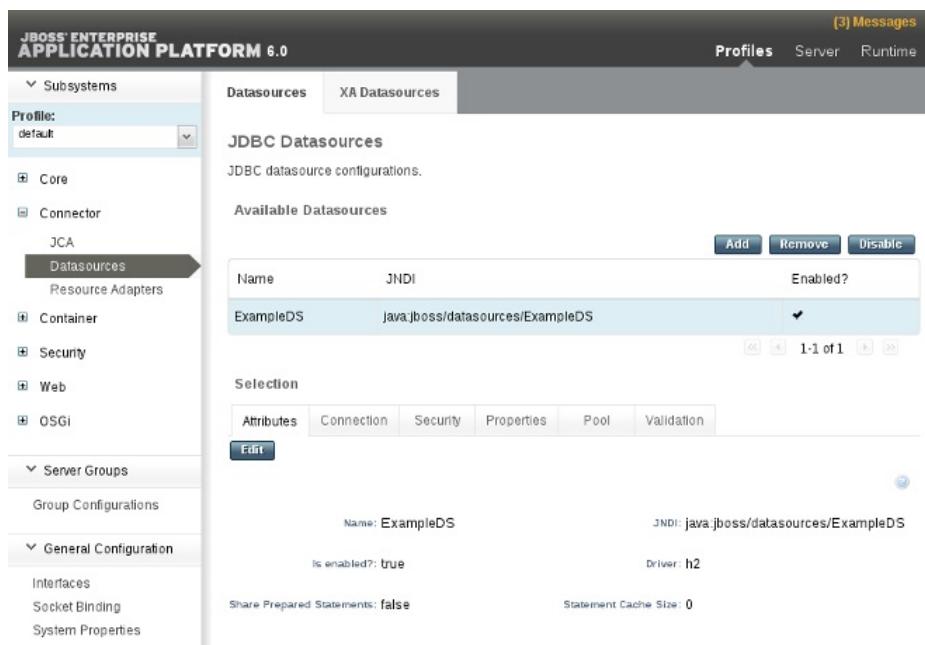


Figure 6.5. Datasources panel

3. Select the XA Datasource panel.

4. Edit the datasource

- Select the relevant XA datasource from the **Available XA Datasources** list. The XA datasource attributes are displayed in the **Attributes** panel below it.
- Select the **Edit** button to edit the datasource attributes.
- Edit the XA datasource attributes and select the **Save** button when done.

Result

The XA datasource has been configured. The changes are now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

- » To create a new datasource, refer here: [Section 6.4.1, “Create an XA Datasource with the Management Interfaces”.](#)

- To remove the datasource, refer here: [Section 6.4.3, “Remove an XA Datasource with the Management Interfaces”.](#)

[Report a bug](#)

6.4.3. Remove an XA Datasource with the Management Interfaces

Task Summary

This topic covers the steps required to remove an XA datasource from JBoss Enterprise Application Platform 6, using either the Management Console or the Management CLI.

Prerequisites

- [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”.](#)

Procedure 6.9. Task

A. Management CLI

- [Section 3.3.2, “Launch the Management CLI”.](#)
- Run the following command to remove an XA datasource:

```
xa-data-source remove --name=XA_DATASOURCE_NAME
```

B. Management Console

- [Section 3.2.2, “Log in to the Management Console”.](#)
- Navigate to the Datasources panel in the Management Console**

a. A. Standalone Mode

Select the **Profile** tab from the top-right of the console.

B. Domain Mode

- Select the **Profiles** tab from the top-right of the console.
- Select the appropriate profile from the drop-down box in the top left.
- Expand the **Subsystems** menu on the left of the console.

b. Select **Connector** → **Datasources** from the menu on the left of the console.

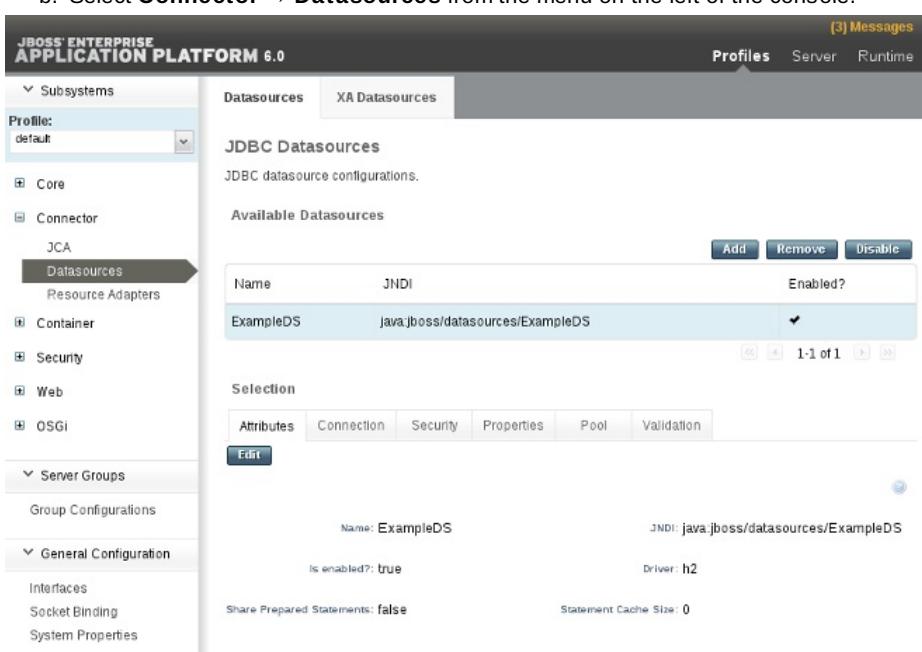


Figure 6.6. Datasources panel

- Select the **XADatasource** panel.
- Select the registered XA datasource to be deleted, and click the **Remove** button in the top right corner of the console.

Result

The XA datasource has been removed from the server.

- » To add a new XA datasource, refer here: [Section 6.4.1, “Create an XA Datasource with the Management Interfaces”](#).

[Report a bug](#)

6.4.4. XA Recovery

6.4.4.1. About XA Recovery Modules

Each XA resource needs a recovery module associated with its configuration. The recovery module must extend class `com.arjuna.ats.jta.recovery.XAResourceRecovery`.

JBoss Enterprise Application Platform provides recovery modules for JDBC and JMS XA resources. For these types of resources, recovery modules are automatically registered. If you need to use a custom module, you can register it in your datasource.

[Report a bug](#)

6.4.4.2. Configure XA Recovery Modules

For most JDBC and JMS 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.

For custom resources which are not JDBC or JMS, contact Red Hat Global Support Services for information on supported configurations.

Each of these configuration attributes can be set either during datasource creation, or afterward. You can set them using either the web-based Management Console or the command-line Management CLI. Refer to [Section 6.4.1, “Create an XA Datasource with the Management Interfaces”](#) and [Section 6.4.2, “Modify an XA Datasource with the Management Interfaces”](#) for general information on configuring XA datasources.

Refer to the following tables for general datasource configuration attributes, and for information about configuration details relating to specific database vendors.

Table 6.2. General Configuration Attributes

Attribute	Description
recovery-username	The username the recovery module should use to connect to the resource for recovery.
recovery-password	The password the recovery module should use to connect to the resource for recovery.
recovery-security-domain	The security domain the recovery module should 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 <code>com.arjuna.ats.jta.recovery.XAResourceRecovery</code> .
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 <code>key=value</code> pairs for the properties.

Vendor-Specific Configuration Information

Oracle

If the Oracle datasource is configured incorrectly, you may see errors like the following in your log output:

```
WARN [com.arjuna.ats.logging.loggerI18N]
[com.arjuna.ats.internal.jta.recovery.xarecovery1] Local
XARecoveryModule.xaRecovery got XA exception
javax.transaction.xa.XAException, XAException.XAER_RMERR
```

To resolve this error, ensure that the Oracle user configured in **recovery-username** has access to the tables needed for recovery. The following SQL statement shows the correct grants for Oracle 11g or Oracle 10g R2 instances patched for Oracle bug 5945463.

```
GRANT SELECT ON sys.dba_pending_transactions TO recovery-username;
GRANT SELECT ON sys.pending_trans$ TO recovery-username;
GRANT SELECT ON sys.dba_2pc_pending TO recovery-username;
GRANT EXECUTE ON sys.dbms_xa TO recovery-username;
```

If you use an Oracle 11 version prior to 11g, change the final **EXECUTE** statement to the following:

```
GRANT EXECUTE ON sys.dbms_system TO recovery-username;
```

PostgreSQL

See the PostgreSQL documentation for instructions on enabling prepared (i.e. XA) transactions. Version 8.4-701 of PostgreSQL's JDBC driver has a bug in **org.postgresql.xa.PGXAConnection** which breaks recovery in certain situations. This is fixed in newer versions.

MySQL

Based on <http://bugs.mysql.com/bug.php?id=12161>, XA transaction recovery did not work in some versions of MySQL 5. This is addressed in MySQL 6.1. Refer to the bug URL or to the MySQL documentation for more information.

IBM DB2

IBM DB2 expects method **XAResource.recover** to be called only during the designated resynchronization stage which occurs when the application server is restarted after a crash or failure. This is a design decision in the DB2 implementation, and out of the scope of this documentation.

[Report a bug](#)

6.5. Datasource Security

6.5.1. About Datasource Security

The preferred solution for datasource security is the use of either security domains or password vaults. Examples of each are included below. For more information, refer to:

- ▶ Security domains: [Section 9.6.1, “About Security Domains”](#).
- ▶ Password vaults: [Section 9.11.1, “About Securing Sensitive Strings in Clear-Text Files”](#).

Example 6.6. Security Domain Example

```
<security>
<security-domain>mySecurityDomain</security-domain>
</security>
```

Example 6.7. Password Vault Example

```
<security>
  <user-name>admin</user-name>

  <password>${VAULT::ds_ExampleDS::password::N2NhZDYz0TMtNWE00S00ZGQ0LWE4MmEtMWN1MDMyNDdmNmI2TE10RV9CUkVBS3ZhdWx0}</password>
</security>
```

[Report a bug](#)

6.6. Datasource Configuration

6.6.1. Datasource Parameters

Table 6.3. Datasource parameters common to non-XA and XA datasources

Parameter	Description
jndi-name	The unique JNDI name for the datasource.
pool-name	The name of the management pool for the datasource.
enabled	Whether or not the datasource is enabled.
use-java-context	Whether to bind the datasource to global JNDI.
spy	Enable spy functionality on the JDBC layer. This logs all JDBC traffic to the datasource. The logging-category parameter must also be set to org.jboss.jdbc .
use-ccm	Enable the cached connection manager.
new-connection-sql	A SQL statement which executes when the connection is added to the connection pool.
transaction-isolation	One of the following: <ul style="list-style-type: none"> » TRANSACTION_READ_UNCOMMITTED » TRANSACTION_READ_COMMITTED » TRANSACTION_REPEATABLE_READ » TRANSACTION_SERIALIZABLE » TRANSACTION_NONE
url-delimiter	The delimiter for URLs in a connection-url for High Availability (HA) clustered databases.
url-selector-strategy-class-name	A class that implements interface org.jboss.jca.adapters.jdbc.URLSelectorStrategy .
security	Contains child elements which are security settings. Refer to Table 6.8, “Security parameters” .
validation	Contains child elements which are validation settings. Refer to Table 6.9, “Validation parameters” .
timeout	Contains child elements which are timeout settings. Refer to Table 6.10, “Timeout parameters” .
statement	Contains child elements which are statement settings. Refer to Table 6.11, “Statement parameters” .

Table 6.4. Non-XA datasource parameters

Parameter	Description
jta	Enable JTA integration for non-XA datasources. Does not apply to XA datasources.
connection-url	The JDBC driver connection URL.
driver-class	The fully-qualified name of the JDBC driver class.
connection-property	Arbitrary connection properties passed to the method <code>Driver.connect(url,props)</code> . Each connection-property specifies a string name/value pair. The property name comes from the name, and the value comes from the element content.
pool	Contains child elements which are pooling settings. Refer to Table 6.6. "Pool parameters common to non-XA and XA datasources" .

Table 6.5. XA datasource parameters

Parameter	Description
xa-datasource-property	A property to assign to implementation class <code>XADatasource</code> . Specified by <code>name=value</code> . If a setter method exists, in the format <code>setName</code> , the property is set by calling a setter method in the format of <code>setName(value)</code> .
xa-datasource-class	The fully-qualified name of the implementation class <code>javax.sql.XADatasource</code> .
driver	A unique reference to the classloader module which contains the JDBC driver. The accepted format is <code>driverName#majorVersion.minorVersion</code> .
xa-pool	Contains child elements which are pooling settings. Refer to Table 6.6. "Pool parameters common to non-XA and XA datasources" and Table 6.7. "XA pool parameters" .
recovery	Contains child elements which are recovery settings. Refer to Table 6.12. "Recovery parameters" .

Table 6.6. Pool parameters common to non-XA and XA datasources

Parameter	Description
min-pool-size	The minimum number of connections a pool holds.
max-pool-size	The maximum number of connections a pool can hold.
prefill	Whether to try to prefill the connection pool. An empty element denotes a true value. The default is false .
use-strict-min	Whether the pool-size is strict. Defaults to false .
flush-strategy	Whether the pool should be flushed in the case of an error. Valid values are: <ul style="list-style-type: none"> » FailingConnectionOnly » IdleConnections » EntirePool The default is FailingConnectionOnly .
allow-multiple-users	Specifies if multiple users will access the datasource through the <code>getConnection(user, password)</code> method, and whether the internal pool type should account for this behavior.

Table 6.7. XA pool parameters

Parameter	Description
is-same-rm-override	Whether the <code>javax.transaction.xa.XAResource.isSameRM(XAResource)</code> class returns true or false .
interleaving	Whether to enable interleaving for XA connection factories.
no-tx-separate-pools	Whether to create separate sub-pools for each context. This is required for Oracle datasources, which do not allow XA connections to be used both inside and outside of a JTA transaction.
pad-xid	Whether to pad the Xid.
wrap-xa-resource	Whether to wrap the XAResource in an <code>org.jboss.tm.XAResourceWrapper</code> instance.

Table 6.8. Security parameters

Parameter	Description
user-name	The username to use to create a new connection.
password	The password to use to create a new connection.
security-domain	Contains the name of a JAAS security-manager which handles authentication. This name correlates to the application-policy/name attribute of the JAAS login configuration.
reauth-plugin	Defines a reauthentication plugin to use to reauthenticate physical connections.

Table 6.9. Validation parameters

Parameter	Description
valid-connection-checker	An implementation of interface <code>org.jboss.jca.adapters.jdbc.ValidConnectionChecker</code> which provides a <code>SQLException.isValidConnection(Connection e)</code> method to validate a connection. An exception means the connection is destroyed. This overrides the parameter <code>check-valid-connection-sql</code> if it is present.
check-valid-connection-sql	An SQL statement to check validity of a pool connection. This may be called when a managed connection is taken from a pool for use.
validate-on-match	Indicates whether connection level validation is performed when a connection factory attempts to match a managed connection for a given set. Mutually exclusive to background-validation .
background-validation	Specifies that connections are validated on a background thread, rather than being validated prior to use. Mutually exclusive to validate-on-match .
background-validation-millis	The amount of time, in milliseconds, that background validation runs.
use-fast-fail	If true, fail a connection allocation on the first attempt, if the connection is invalid. Defaults to false .
stale-connection-checker	An instance of <code>org.jboss.jca.adapters.jdbc.StaleConnectionChecker</code> which provides a Boolean <code>isStaleConnection(SQLException e)</code> method. If this method returns <code>true</code> , the exception is wrapped in an <code>org.jboss.jca.adapters.jdbc.StaleConnectionException</code> , which is a subclass of <code>SQLException</code> .
exception-sorter	An instance of <code>org.jboss.jca.adapters.jdbc.ExceptionSorter</code> which provides a Boolean <code>isExceptionFatal(SQLException e)</code> method. This method validates whether an exception should be broadcast to all instances of <code>javax.resource.spi.ConnectionEventListener</code> as a <code>connectionErrorOccurred</code> message.

Table 6.10. Timeout parameters

Parameter	Description
blocking-timeout-millis	The maximum time, in milliseconds, to block while waiting for a connection. After this time is exceeded, and exception is thrown. This blocks only while waiting for a permit for a connection, and does not throw an exception if creating a new connection takes a long time. Defaults to 30000, which is 30 seconds.
idle-timeout-minutes	The maximum time, in minutes, before an idle connection is closed. The actual maximum time depends upon the idleRemover scan time, which is half of the smallest idle-timeout-minutes of any pool.
set-tx-query-timeout	Whether to set the query timeout based on the time remaining until transaction timeout. Any configured query timeout is used if no transaction exists. Defaults to false .
query-timeout	Timeout for queries, in seconds. The default is no timeout.
allocation-retry	The number of times to retry allocating a connection before throwing an exception. The default is 0 , so an exception is thrown upon the first failure.
allocation-retry-wait-millis	How long, in milliseconds, to wait before retrying to allocate a connection. The default is 5000, which is 5 seconds.
xa-resource-timeout	If non-zero, this value is passed to method XAResource.setTransactionTimeout .

Table 6.11. Statement parameters

Parameter	Description
track-statements	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. This is the default.
prepared-statement-cache-size	The number of prepared statements per connection, in a Least Recently Used (LRU) cache.
share-prepared-statements	Whether asking for the same statement twice without closing it uses the same underlying prepared statement. The default is false .

Table 6.12. Recovery parameters

Parameter	Description
recover-credential	A username/password pair or security domain to use for recovery.
recover-plugin	An implementation of class org.jboss.jca.core.spi.recoveryRecoveryPlugin class, to be used for recovery.

[Report a bug](#)

6.6.2. Datasource Connection URLs

Table 6.13.

Datasource	Connection URL
PostgreSQL	<code>jdbc:postgresql://SERVER_NAME:PORT/DATABASE_NAME</code>
MySQL	<code>jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME</code>
Oracle	<code>jdbc:oracle:thin:@ORACLE_HOST:PORT:ORACLE_SID</code>
IBM DB2	<code>jdbc:db2://SERVER_NAME:PORT/DATABASE_NAME</code>
Microsoft SQLServer	<code>jdbc:microsoft:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME</code>

[Report a bug](#)

6.6.3. Datasource Extensions

Datasource deployments can use several extensions in the JDBC resource adapter to improve the connection validation, and check whether an exception should reestablish the connection. Those extensions are:

Table 6.14. Datasource Extensions

Datasource Extension	Configuration Parameter	Description
<code>org.jboss.jca.adapters.jdbc.spi.ExceptionSorter</code>	<code><exception-sorter></code>	Checks whether an SQLException is fatal for the connection on which it was thrown
<code>org.jboss.jca.adapters.jdbc.spi.StaleConnection</code>	<code><stale-connection-checker></code>	Wraps stale SQLExceptions in a <code>org.jboss.jca.adapters.jdbc.StaleConnectionException</code>
<code>org.jboss.jca.adapters.jdbc.spi.ValidConnection</code>	<code><valid-connection-checker></code>	Checks whether a connection is valid for use by the application

JBoss Enterprise Application Platform 6 also features implementations of these extensions for several supported databases.

Extension Implementations

Generic

- » `org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter`
- » `org.jboss.jca.adapters.jdbc.extensions.novendor.NullStaleConnectionChecker`
- » `org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker`
- » `org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker`

PostgreSQL

- » `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter`
- » `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker`

MySQL

- » `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter`
- » `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker`
- » `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker`

IBM DB2

- » org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter
- » org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker
- » org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker

Sybase

- » org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter
- » org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker

Microsoft SQLServer

- » org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker

Oracle

- » org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter
- » org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter
- » org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker

[Report a bug](#)

6.7. Example Datasources

6.7.1. Example PostgreSQL Datasource

Example 6.8.

The example below is a PostgreSQL datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<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.PostgreSQLValidConnectionChe
cker"></valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter">
    </exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="postgresql" module="org.postgresql">
      <xa-datasource-class>org.postgresql.xa.PGXDataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the PostgreSQL datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.2. Example PostgreSQL XA Datasource

Example 6.9.

The example below is a PostgreSQL XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <xa-datasource jndi-name="java:jboss/PostgresXADS" pool-name="PostgresXADS">
    <driver>postgresql</driver>
    <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>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChe
cker">
        </valid-connection-checker>
        <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter">
          </exception-sorter>
      </validation>
    </xa-datasource>
    <drivers>
      <driver name="postgresql" module="org.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXDataSource</xa-datasource-class>
      </driver>
    </drivers>
  </datasources>
```

The example below is a **module.xml** file for the PostgreSQL XA datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.3. Example MySQL Datasource

Example 6.10.

The example below is a MySQL datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <datasource jndi-name="java:jboss/MySQLDS" pool-name="MySQLDS">
    <connection-url>jdbc:mysql://mysql-localhost:3306/jbosssdb</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"><
/valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"></except
ion-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the MySQL datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.4. Example MySQL XA Datasource

Example 6.11.

The example below is a MySQL XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
<xa-datasource jndi-name="java:jboss/MysqlXADS" pool-name="MysqlXADS">
<driver>mysql</driver>
<xa-datasource-property name="ServerName">localhost</xa-datasource-property>
<xa-datasource-property name="DatabaseName">mysqlDb</xa-datasource-property>
<security>
<user-name>admin</user-name>
<password>admin</password>
</security>
<validation>
<valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"><
/>
<valid-connection-checker>
<exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"></except
ion-sorter>
</validation>
</xa-datasource>
<drivers>
<driver name="mysql" module="com.mysql">
<xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-
datasource-class>
</driver>
</drivers>
</datasources>
```

The example below is a **module.xml** file for the MySQL XA datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
<resources>
<resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
</resources>
<dependencies>
<module name="javax.api"/>
<module name="javax.transaction.api"/>
</dependencies>
</module>
```

[Report a bug](#)

6.7.5. Example Oracle Datasource



Oracle Datasources

Prior to version 10.2 of the Oracle datasource, the <no-tx-separate-pools/> parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.

Example 6.12.

The example below is an Oracle datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <datasource jndi-name="java:/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"
      ></valid-connection-checker>
      <stale-connection-checker class-
        name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"
      ></stale-connection-checker>
      <exception-sorter class-
        name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"></exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
      <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-
      datasource-class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the Oracle datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.6. Example Oracle XA Datasource



Oracle Datasources

Prior to version 10.2 of the Oracle datasource, the `<no-tx-separate-pools/>` parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.



Important

The following settings must be applied for the user accessing an Oracle XA datasource in order for XA recovery to operate correctly:

- ▷ 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;

Example 6.13.

The example below is an Oracle XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <xa-datasource jndi-name="java:/XAOracleDS" pool-name="XAOracleDS">
    <driver>oracle</driver>
    <xa-datasource-property name="URL">jdbc:oracle:oci8:@tc</xa-datasource-
property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
      <no-tx-separate-pools />
    </xa-pool>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"
></valid-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"
></stale-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"></exce
ption-sorter>
      </validation>
    </xa-datasource>
    <drivers>
      <driver name="oracle" module="com.oracle">
        <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-
datasource-class>
      </driver>
    </drivers>
  </datasources>
```

The example below is a **module.xml** file for the Oracle XA datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.7. Example Microsoft SQLServer Datasource

Example 6.14.

The example below is a Microsoft SQLServer datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <datasource jndi-name="java:/MSSQLDS" pool-name="MSSQLDS">
    <connection-
url>jdbc:microsoft: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"><
/valid-connection-checker>
    </validation>
  </datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-
class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the Microsoft SQLServer datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.8. Example Microsoft SQLServer XA Datasource

Example 6.15.

The example below is a Microsoft SQLServer XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
<xa-datasource jndi-name="java:/MSSQLXADS" pool-name="MSSQLXADS">
<driver>sqlserver</driver>
<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>
<security>
<user-name>admin</user-name>
<password>admin</password>
</security>
<xa-pool>
<is-same-rm-override>false</is-same-rm-override>
</xa-pool>
<validation>
<valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"><
/>
</valid-connection-checker>
</validation>
</xa-datasource>
<drivers>
<driver name="sqlserver" module="com.microsoft">
<xa-datasource-
class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-class>
</driver>
</drivers>
</datasources>
```

The example below is a **module.xml** file for the Microsoft SQLServer XA datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
<resources>
<resource-root path="sqljdbc4.jar"/>
</resources>
<dependencies>
<module name="javax.api"/>
<module name="javax.transaction.api"/>
</dependencies>
</module>
```

[Report a bug](#)

6.7.9. Example IBM DB2 Datasource

Example 6.16.

The example below is an IBM DB2 datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <datasource jndi-name="java:/DB2DS" pool-name="DB2DS">
    <connection-url>jdbc:db2: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"></val
id-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker"></stal
e-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></exception-
sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
      <xa-datasource-class>com.ibm.jdbc.DB2XADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the IBM DB2 datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.10. Example IBM DB2 XA Datasource

Example 6.17.

The example below is an IBM DB2 XA datasource configuration. The datasource has been enabled, a user has been added and validation options have been set.

```
<datasources>
  <xa-datasource jndi-name="java:/DB2XADS" pool-name="DB2XADS">
    <driver>ibmdb2</driver>
    <xa-datasource-property name="DatabaseName">ibmdb2db</xa-datasource-
property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"></val
id-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker"></stal
e-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></exception-
sorter>
    </validation>
    <recovery>
      <recovery-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>
      </recovery-plugin>
    </recovery>
  </xa-datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
      <xa-datasource-class>com.ibm.db2.jdbc.DB2XADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the IBM DB2 XA datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.11. Example Sybase Datasource

Example 6.18.

The example below is a Sybase datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB"
  enabled="true">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?
JCONNECT_VERSION=6</connection-url>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"></exe
ption-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <datasource-class>com.sybase.jdbc2.jdbc.SybDataSource</datasource-class>
      <xa-datasource-class>com.sybase.jdbc3.jdbc.SybXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the Sybase datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.7.12. Example Sybase XA Datasource

Example 6.19.

The example below is a Sybase XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

```
<datasources>
  <xa-datasource jndi-name="java:jboss/SybaseXADS" pool-name="SybaseXADS"
    enabled="true">
    <xa-datasource-property name="NetworkProtocol">Tds</xa-datasource-property>
    <xa-datasource-property name="ServerName">myserver</xa-datasource-property>
    <xa-datasource-property name="PortNumber">4100</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">mydatabase</xa-datasource-
    property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
      name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker">
    </valid-connection-checker>
      <exception-sorter class-
      name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"></exce
      ption-sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <datasource-class>com.sybase.jdbc2.jdbc.SybDataSource</datasource-class>
      <xa-datasource-class>com.sybase.jdbc3.jdbc.SybXADataSource</xa-datasource-
      class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the Sybase XA datasource above.

```
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

Chapter 7. Configuring Modules

7.1. Introduction

7.1.1. Modules

A Module is a logical grouping of classes used for class loading and dependency management. JBoss Enterprise Application Platform 6 identifies two different types of modules, sometimes called static and dynamic modules. However the only difference between the two is how they are packaged. All modules provide the same features.

Static Modules

Static Modules are predefined in the `EAP_HOME/modules/` directory of the application server. Each sub-directory represents one module and contains one or more JAR files and a configuration file (`module.xml`). The name of the module is defined in the `module.xml` file. All the application server provided APIs are provided as static modules, including the Java EE APIs as well as other APIs such as JBoss Logging.

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 the JBoss administrator. The applications can then declare an explicit dependency on the custom static modules.

Dynamic Modules

Dynamic Modules are created and loaded by the application server for each JAR or WAR deployment (or 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.

[Report a bug](#)

7.1.2. Global Modules

A global module is a module that JBoss Enterprise Application Platform 6 provides as a dependency to every application. Any module can be made global by adding it to the application server's list of global modules. It does not require changes to the module.

[Report a bug](#)

7.1.3. Module Dependencies

A module dependency is a declaration that one module requires the classes of another module in order to function. Modules can declare dependencies on any number of other modules. When the application server 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.

Deployed applications (JAR and WAR) are loaded as dynamic modules and make use of dependencies to access the APIs provided by JBoss Enterprise Application Platform 6.

There are two types of dependencies: explicit and implicit.

Explicit dependencies are declared in configuration by the developer. Static modules can declare dependencies in the `modules.xml` file. Dynamic modules can have dependencies declared in the `MANIFEST.MF` or `jboss-deployment-structure.xml` deployment descriptors of the deployment.

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.

Implicit dependencies are added automatically by the application server when certain conditions or metadata are found in a deployment. The Java EE 6 APIs supplied with JBoss Enterprise Application Platform

are examples of modules that are added by detection of implicit dependencies in deployments.

Deployments can also be configured to exclude specific implicit dependencies. This is done with the `jboss-deployment-structure.xml` deployment descriptor file. This is commonly done when an application bundles a specific version of a library that the application server will attempt to add as an implicit 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.

Example 7.1. Module dependencies

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.

[Report a bug](#)

7.1.4. Subdeployment Class Loader Isolation

Each subdeployment in an Enterprise Archive (EAR) is a dynamic module with its own class loader and cannot access the resources of other subdeployments. This is called subdeployment class loader isolation.

JBoss Enterprise Application Platform 6 has strict subdeployment class loader isolation disabled by default. It can be enabled if required.

[Report a bug](#)

7.2. Disable Sub-Deployment Module Isolation for All Deployments

This task shows server administrators how to disable Subdeployment Module Isolation on the application server. This affects all deployments.



Warning

This task requires you to edit the XML configuration files of the server. The server must be halted before doing this. This is temporary as the final release administration tools will support this type of configuration.

1. Stop the server

Halt the JBoss Enterprise Application Platform server.

2. Open the server configuration file

Open the server configuration file in a text editor.

This file will be different for a managed domain or standalone server. In addition, non-default locations and file names may be used. The default configuration files are

`domain/configuration/domain.xml` and

`standalone/configuration/standalone.xml` for managed domains and standalone servers respectively.

3. Locate the EE Subsystem Configuration

Locate the EE Subsystem configuration element in the configuration file. The `<profile>` element of the configuration file contains several subsystem elements. The EE Subsystem element has the namespace of `urn:jboss:domain:ee:1.0`.

```
<profile>
  ...
  <subsystem xmlns="urn:jboss:domain:ee:1.0" />
  ...

```

The default configuration has a single self-closing tag but a custom configuration may have separate open and closing tags (possibly with other elements within) like this:

```
<subsystem xmlns="urn:jboss:domain:ee:1.0" ></subsystem>
```

4. Replace self-closing tags if necessary

If the EE Subsystem element is a single self-closing tag then replace with with appropriate opening and closing tags like this:

```
<subsystem xmlns="urn:jboss:domain:ee:1.0" ></subsystem>
```

5. Add ear-subdeployments-isolated element

Add the **ear-subdeployments-isolated** element as a child of the EE Subsystem element and add the content of **false** like this:

```
<subsystem xmlns="urn:jboss:domain:ee:1.0" ><ear-subdeployments-isolated>false</ear-subdeployments-isolated></subsystem>
```

6. Start the server

Relaunch the JBoss Enterprise Application Platform server to start it running with the new configuration.

Result:

The server will now be running with Subdeployment Module Isolation disabled for all deployments.

[Report a bug](#)

7.3. Add a module to all deployments

This task shows how JBoss administrators can define a list of global modules.

Prerequisites

1. You must know the name of the modules that are to be configured as global modules. Refer to [Section 7.4.1, “Included Modules”](#) for the list of static modules included with JBoss Enterprise Application Platform 6. If the module is in another deployment, refer to [Section 7.4.2, “Dynamic Module Naming”](#) to determine the module name.

Procedure 7.1. Add a module to the list of global modules

1. Login to the management console. [Section 3.2.2, “Log in to the Management Console”](#)
2. Navigate to the **EE Subsystem** panel.

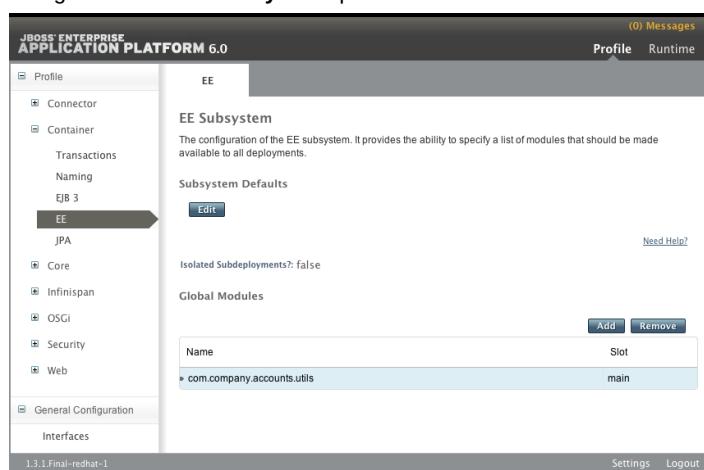


Figure 7.1. EE Subsystem Panel

3. Click the **Add** button in the **Global Modules** section. The **Create Module** dialog appears.
4. Type in the name of the module and optionally the module slot.
5. Click the **Save** button to add the new global module, or click the **Cancel** link to abort.
 - ▶ If you click the **Save** button, the dialog will close and the specified module will be added to the list of global modules.
 - ▶ If you click **Cancel**, the dialog will close and no changes will be made.

Result

The modules added to the list of global modules will be added as dependencies to every deployment.

[Report a bug](#)

7.4. Reference

7.4.1. Included Modules

- ▶ **asm.asm**
- ▶ **ch.qos.cal10n**
- ▶ **com.google.guava**
- ▶ **com.h2database.h2**
- ▶ **com.sun.jsf-impl**
- ▶ **com.sun.jsf-impl**
- ▶ **com.sun.xml.bind**
- ▶ **com.sun.xml.messaging.saaj**
- ▶ **gnu getopt**
- ▶ **javaee.api**
- ▶ **javax.activation.api**
- ▶ **javax.annotation.api**
- ▶ **javax.api**
- ▶ **javax.ejb.api**
- ▶ **javax.el.api**
- ▶ **javax.enterprise.api**
- ▶ **javax.enterprise.deploy.api**
- ▶ **javax.faces.api**
- ▶ **javax.faces.api**
- ▶ **javax.inject.api**
- ▶ **javax.interceptor.api**
- ▶ **javax.jms.api**
- ▶ **javax.jws.api**
- ▶ **javax.mail.api**
- ▶ **javax.management.j2ee.api**
- ▶ **javax.persistence.api**
- ▶ **javax.resource.api**
- ▶ **javax.rmi.api**
- ▶ **javax.security.auth.message.api**
- ▶ **javax.security.jacc.api**
- ▶ **javax.servlet.api**
- ▶ **javax.servlet.jsp.api**
- ▶ **javax.servlet.jstl.api**
- ▶ **javax.transaction.api**
- ▶ **javax.validation.api**

- » javax.ws.rs.api
- » javax.wsdl4j.api
- » javax.xml.bind.api
- » javax.xml.jaxp-provider
- » javax.xml.registry.api
- » javax.xml.rpc.api
- » javax.xml.soap.api
- » javax.xml.stream.api
- » javax.xml.ws.api
- » jline
- » net.sourceforge.cssparser
- » net.sourceforge.htmlunit
- » net.sourceforge.nekohtml
- » nu.xom
- » org.antlr
- » org.apache.ant
- » org.apache.commons.beanutils
- » org.apache.commons.cli
- » org.apache.commons.codec
- » org.apache.commons.collections
- » org.apache.commons.io
- » org.apache.commons.lang
- » org.apache.commons.logging
- » org.apache.commons.pool
- » org.apache.cxf
- » org.apache.httpcomponents
- » org.apache.james.mime4j
- » org.apache.log4j
- » org.apache.neethi
- » org.apache.santuario.xmlsec
- » org.apache.velocity
- » org.apache.ws.scout
- » org.apache.ws.security
- » org.apache.ws.xmlschema
- » org.apache.xalan
- » org.apache.xerces
- » org.apache.xml-resolver
- » org.codehaus.jackson.jackson-core-asl
- » org.codehaus.jackson.jackson-jaxrs
- » org.codehaus.jackson.jackson-mapper-asl
- » org.codehaus.jackson.jackson-xc
- » org.codehaus.woodstox
- » org.dom4j
- » org.hibernate
- » org.hibernate.envers
- » org.hibernate.infinispan
- » org.hibernate.validator
- » org.hornetq
- » org.hornetq.ra
- » org.infinispan
- » org.infinispan.cacheStore.jdbc
- » org.infinispan.cacheStore.remote
- » org.infinispan.client.hotrod

» org.jacorb
» org.javassist
» org.jaxen
» org.jboss.as.aggregate
» org.jboss.as.appclient
» org.jboss.as.cli
» org.jboss.as.clustering.api
» org.jboss.as.clustering.common
» org.jboss.as.clustering.ejb3.infinispan
» org.jboss.as.clustering.impl
» org.jboss.as.clustering.infinispan
» org.jboss.as.clustering.jgroups
» org.jboss.as.clustering.service
» org.jboss.as.clustering.singleton
» org.jboss.as.clustering.web.infinispan
» org.jboss.as.clustering.web.spi
» org.jboss.as.cmp
» org.jboss.as.connector
» org.jboss.as.console
» org.jboss.as.controller
» org.jboss.as.controller-client
» org.jboss.as.deployment-repository
» org.jboss.as.deployment-scanner
» org.jboss.as.domain-add-user
» org.jboss.as.domain-http-error-context
» org.jboss.as.domain-http-interface
» org.jboss.as.domain-management
» org.jboss.as.ee
» org.jboss.as.ee.deployment
» org.jboss.as.ejb3
» org.jboss.as.embedded
» org.jboss.as.host-controller
» org.jboss.as.jacorb
» org.jboss.as.jaxr
» org.jboss.as.jaxrs
» org.jboss.as.jdr
» org.jboss.as.jmx
» org.jboss.as.jpa
» org.jboss.as.jpa.hibernate
» org.jboss.as.jpa.hibernate
» org.jboss.as.jpa.hibernate.infinispan
» org.jboss.as.jpa.openjpa
» org.jboss.as.jpa.spi
» org.jboss.as.jpa.util
» org.jboss.as.jsr77
» org.jboss.as.logging
» org.jboss.as.mail
» org.jboss.as.management-client-content
» org.jboss.as.messaging
» org.jboss.as.modcluster
» org.jboss.as.naming
» org.jboss.as.network
» org.jboss.as.osgi

- » **org.jboss.as.platform-mbean**
- » **org.jboss.as.pojo**
- » **org.jboss.as.process-controller**
- » **org.jboss.as.protocol**
- » **org.jboss.as.remoting**
- » **org.jboss.as.sar**
- » **org.jboss.as.security**
- » **org.jboss.as.server**
- » **org.jboss.as.standalone**
- » **org.jboss.as.threads**
- » **org.jboss.as.transactions**
- » **org.jboss.as.web**
- » **org.jboss.as.webservices**
- » **org.jboss.as.webservices.server.integration**
- » **org.jboss.as.webservices.server.jaxrpc-integration**
- » **org.jboss.as.weld**
- » **org.jboss.as.xts**
- » **org.jboss.classfilewriter**
- » **org.jboss.com.sun.httpserver**
- » **org.jboss.common-core**
- » **org.jboss.dmr**
- » **org.jboss.ejb-client**
- » **org.jboss.ejb3**
- » **org.jboss.iiop-client**
- » **org.jboss.integration.ext-content**
- » **org.jboss.interceptor**
- » **org.jboss.interceptor.spi**
- » **org.jboss.invocation**
- » **org.jboss.ironjacamar.api**
- » **org.jboss.ironjacamar.impl**
- » **org.jboss.ironjacamar.jdbcadapters**
- » **org.jboss.jandex**
- » **org.jboss.jaxbintros**
- » **org.jboss.jboss-transaction-spi**
- » **org.jboss.jsfunit.core**
- » **org.jboss.jts**
- » **org.jboss.jts.integration**
- » **org.jboss.logging**
- » **org.jboss.logmanager**
- » **org.jboss.logmanager.log4j**
- » **org.jboss.marshalling**
- » **org.jboss.marshalling.river**
- » **org.jboss.metadata**
- » **org.jboss.modules**
- » **org.jboss.msc**
- » **org.jboss.netty**
- » **org.jboss.osgi.deployment**
- » **org.jboss.osgi.framework**
- » **org.jboss.osgi.resolver**
- » **org.jboss.osgi.spi**
- » **org.jboss.osgi.vfs**
- » **org.jboss.remoting3**

```
▶ org.jboss.resteasy.resteasy-atom-provider
▶ org.jboss.resteasy.resteasy-cdi
▶ org.jboss.resteasy.resteasy-jackson-provider
▶ org.jboss.resteasy.resteasy-jaxb-provider
▶ org.jboss.resteasy.resteasy-jaxrs
▶ org.jboss.resteasy.resteasy-jsapi
▶ org.jboss.resteasy.resteasy-multipart-provider
▶ org.jboss.sasl
▶ org.jboss.security.negotiation
▶ org.jboss.security.xacml
▶ org.jboss.shrinkwrap.core
▶ org.jboss.staxmapper
▶ org.jboss.stdio
▶ org.jboss.threads
▶ org.jboss.vfs
▶ org.jboss.weld.api
▶ org.jboss.weld.core
▶ org.jboss.weld.spi
▶ org.jboss.ws.api
▶ org.jboss.ws.common
▶ org.jboss.ws.cxf.jbossws-cxf-client
▶ org.jboss.ws.cxf.jbossws-cxf-factories
▶ org.jboss.ws.cxf.jbossws-cxf-server
▶ org.jboss.ws.cxf.jbossws-cxf-transports-httpserver
▶ org.jboss.ws.jaxws-client
▶ org.jboss.ws.jaxws-jboss-httpserver-httpspi
▶ org.jboss.ws.native.jbossws-native-core
▶ org.jboss.ws.native.jbossws-native-factories
▶ org.jboss.ws.native.jbossws-native-services
▶ org.jboss.ws.saaj-impl
▶ org.jboss.ws.spi
▶ org.jboss.ws.tools.common
▶ org.jboss.ws.tools.wsconsume
▶ org.jboss.ws.tools.wsprovide
▶ org.jboss.xb
▶ org.jboss.xnio
▶ org.jboss.xnio.nio
▶ org.jboss.xts
▶ org.jdom
▶ org.jgroups
▶ org.joda.time
▶ org.junit
▶ org.omg.api
▶ org.osgi.core
▶ org.picketbox
▶ org.picketlink
▶ org.python.jython.standalone
▶ org.scannotation.scannotation
▶ org.slf4j
▶ org.slf4j.ext
▶ org.slf4j.impl
▶ org.slf4j.jcl-over-slf4j
▶ org.w3c.css.sac
```

» **sun.jdk**

[Report a bug](#)

7.4.2. Dynamic Module Naming

All deployments are loaded as modules by JBoss Enterprise Application Platform 6 and named according to the following conventions.

1. Deployments of WAR and JAR files are named with 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.

2. Subdeployments within an Enterprise Archive are named with 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.accounting.ear.reports.war**.

[Report a bug](#)

Chapter 8. Application Deployment

8.1. About Application Deployment

JBoss Enterprise Application Platform 6 features a range of application deployment and configuration options to cater to both administrative and development environments. For administrators, the Management Console and the Management CLI offer the ideal graphical and command line interfaces to manage application deployments in a production environment. For developers, the range of application deployment testing options include a highly configurable filesystem deployment scanner, the use of an IDE such as JBoss Developer Studio, or deployment and undeployment via Maven.

Administration

- ▶ **Management Console**

- [Section 8.2.2, “Deploy an Application Using the Management Console”](#)
- [Section 8.2.3, “Undeploy an Application Using the Management Console”](#)

- ▶ **Management CLI**

- [Section 8.3.2, “Deploy an Application in a Managed Domain Using the Management CLI”](#)
- [Section 8.3.4, “Deploy an Application in a Standalone Server Using the Management CLI”](#)
- [Section 8.3.3, “Undeploy an Application in a Managed Domain Using the Management CLI”](#)
- [Section 8.3.5, “Undeploy an Application in a Standalone Server Using the Management CLI”](#)
- [Section 8.3.1, “Manage Application Deployment in the Management CLI”](#)

Development

- ▶ **Deployment Scanner**

- [Section 8.4.7, “Configure the Deployment Scanner”](#)
- [Section 8.4.2, “Deploy an Application to a Standalone Server Instance with the Deployment Scanner”](#)
- [Section 8.4.3, “Undeploy an Application to a Standalone Server Instance with the Deployment Scanner”](#)
- [Section 8.4.4, “Redeploy an Application to a Standalone Server Instance with the Deployment Scanner”](#)
- [Section 8.4.8, “Configure the Deployment Scanner with the Management CLI”](#)
- [Section 8.4.6, “Reference for Deployment Scanner Attributes”](#)
- [Section 8.4.5, “Reference for Deployment Scanner Marker Files”](#)

- ▶ **Maven**

- [Section 8.5.2, “Deploy an Application with Maven”](#)
- [Section 8.5.3, “Undeploy an Application with Maven”](#)

[Report a bug](#)

8.2. Deploy with the Management Console

8.2.1. Manage Application Deployment in the Management Console

Deploying applications via the Management Console gives you the benefit of a graphical interface that is easy to use. You can see at a glance what applications are deployed to your server or server groups, and you can disable or delete applications from the content repository as required.

[Report a bug](#)

8.2.2. Deploy an Application Using the Management Console

Prerequisites

- ▶ [Section 3.2.2, “Log in to the Management Console”](#)
- ▶ [Section 3.2.5, “Add a Deployment in the Management Console”](#)

Procedure 8.1. Task

1. **Navigate to the Manage Deployments panel in the Management Console**
 - a. Select the **Runtime** tab from the top right of the console.

- b. Select the **Deployments** → **Manage Deployments** option from the menu on the left of the console.

2. Deploy an application

The deployment method will differ depending on whether you are deploying to a standalone server instance or a managed domain.

A. Deploy to a standalone server instance

The **Deployments** table shows all available application deployments and their status.

The screenshot shows the JBoss Enterprise Application Platform 6.0 Management Console. The left sidebar has sections for Server Status, Subsystem Metrics (Datasources, JPA, Transactions, Web), Runtime Operations (OSGI), Deployments (Manage Deployments, Webservices), and a navigation bar at the top with Profile and Runtime tabs. The main area is titled 'Deployments' and contains a table with one row:

Name	Runtime Name	Enabled	En/Disable	Remove
your_application.jar	your_application.jar	<input checked="" type="checkbox"/>	<input type="button" value="Enable"/>	<input type="button" value="Remove"/>

At the bottom of the table are navigation buttons: <<, <, 1-1 of 1, >, >>.

Figure 8.1. Available deployments

a. Enable the application in a standalone server instance

Click on the **Enable** button in the **Deployments** table to enable the application deployment.

b. Confirm

Click on the **confirm** button to confirm that the application will be enabled on the server instance.

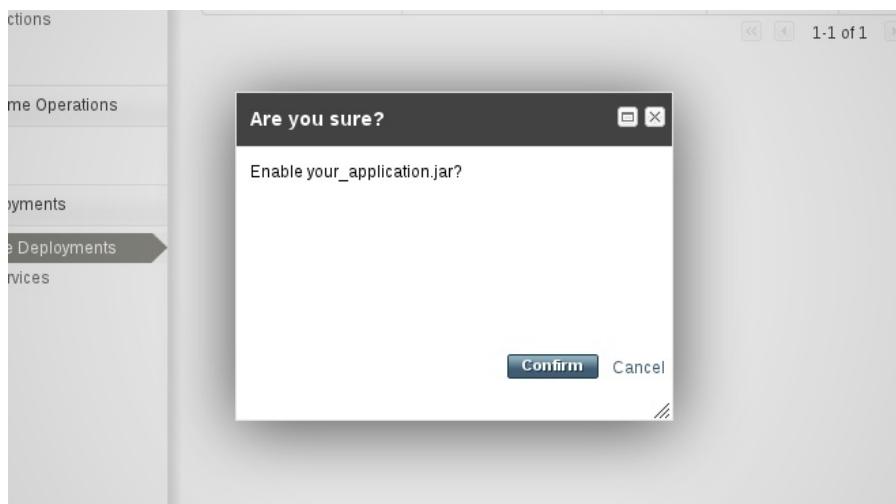


Figure 8.2. Available deployments in a standalone server

B. Deploy to a managed domain

The **Deployment Content** section contains a **Content Repository** table showing all

available application deployments and their status.

The screenshot shows the JBoss Enterprise Application Platform 6.0 interface. The top navigation bar includes tabs for 'Profiles', 'Server', and 'Runtime'. The 'Runtime' tab is selected. On the left, a sidebar menu lists 'Domain Status', 'Subsystem Metrics', 'P runtime Operations', and 'Deployments'. Under 'Deployments', 'Manage Deployments' is selected. The main content area is titled 'Content Repository' and contains a table with one row:

Name	Runtime Name	Add to Groups	Remove
your_application.jar	your_application.jar	<input type="button" value="Add to Groups"/>	<input type="button" value="Remove"/>

Below the table are navigation buttons: <<, <, >, >>, and '1-1 of 1'.

Figure 8.3. Available deployments in a managed domain

a. **Enable the application in a Managed Domain**

Click on the **Add to Groups** button in the **Content Repository** table.

b. **Select server groups**

Check the boxes for each of the server groups that you want the application to be added to and click on the **Save** button to continue.

The screenshot shows a modal dialog box titled 'Select server groups' for the application 'your_application.jar'. The dialog has a header 'Select server groups for your_application.jar'. It contains a table with three rows:

Server Group	Profile	Add to Group
main-server-group	full	<input checked="" type="checkbox"/>
new-server-group	full	<input checked="" type="checkbox"/>
other-server-group	full-ha	<input checked="" type="checkbox"/>

Below the table are navigation buttons: <<, <, >, >>, and '1-3 of 3'. A note 'Enable your_application.jar' is displayed below the table. At the bottom are 'Save' and 'Cancel' buttons.

Figure 8.4. Select server groups for application deployment

c. **Confirm**

Click on the **Server Group Deployments** tab to view the **Server Groups** table.

Your application is now deployed to the server groups that you have selected.

The screenshot shows the JBoss Management Console interface. On the left, there's a sidebar with various monitoring and management tabs like Server Status, Domain Status, Server Instances, JVM Status, Subsystem Metrics, Datasources, JPA, JMS Destinations, Transactions, Web, Runtime Operations (OSGI), and Deployments. The Deployments tab is currently selected. In the main content area, there are two sections: 'Server Groups' and 'Deployments for main-server-group'. The 'Server Groups' section lists three groups: 'main-server-group' (Profile: full), 'new-server-group' (Profile: full), and 'other-server-group' (Profile: full-ha). The 'Deployments for main-server-group' section shows a single deployment named 'your_application.jar' with runtime name 'your_application.jar', enabled status checked, and buttons for 'Disable' and 'Remove'. Navigation buttons at the bottom indicate 1-3 of 3.

Figure 8.5. Confirmation of application deployment to server groups

Result

The application is deployed on the relevant server or server group.

[Report a bug](#)

8.2.3. Undeploy an Application Using the Management Console

Prerequisites

- ▶ [Section 3.2.2, "Log in to the Management Console"](#)
- ▶ [Section 3.2.5, "Add a Deployment in the Management Console"](#)
- ▶ [Section 8.2.2, "Deploy an Application Using the Management Console"](#)

Procedure 8.2. Task

1. Navigate to the Manage Deployments panel in the Management Console

- a. Select the **Runtime** tab from the top right of the console.
- b. Select the **Deployments** → **Manage Deployments** option from the menu on the left of the console.

2. Undeploy an application

The undeployment method will differ depending on whether you are deploying to a standalone server instance or a managed domain.

A. Undeploy from a standalone server instance

The **Deployments** table shows all available application deployments and their status.

Name	Runtime Name	Enabled	En/Disable	Remove
your_application.jar	your_application.jar	<input checked="" type="checkbox"/>	<input type="button" value="Disable"/>	<input type="button" value="Remove"/>

1-1 of 1

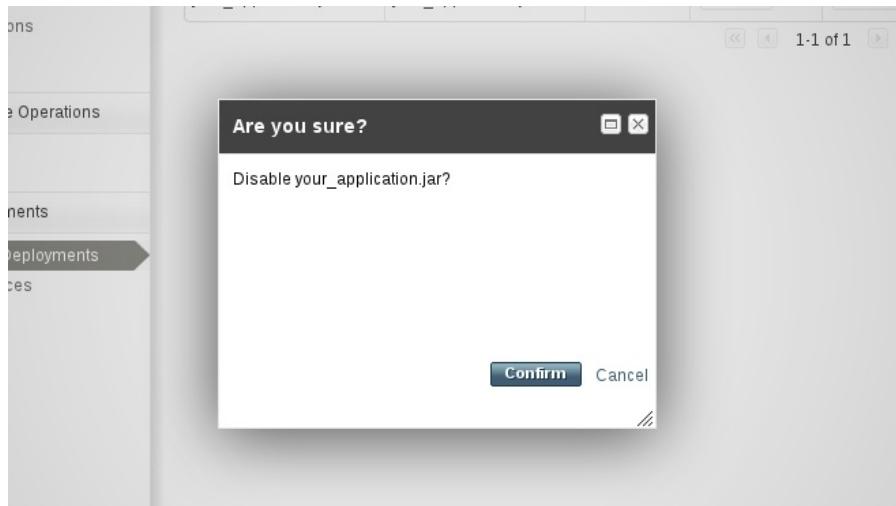
Figure 8.6. Available deployments

a. **Disable the application in a standalone server instance**

Click on the **Disable** button in the **Deployments** table to disable the application.

b. **Confirm that you wish to disable the application**

Click on the **confirm** button to confirm that the application will be disabled on the server instance.

**Figure 8.7. Confirm the application to disable**

B. Undeploy from a managed domain

The **Deployment Content** section contains a **Content Repository** table showing all available application deployments and their status.

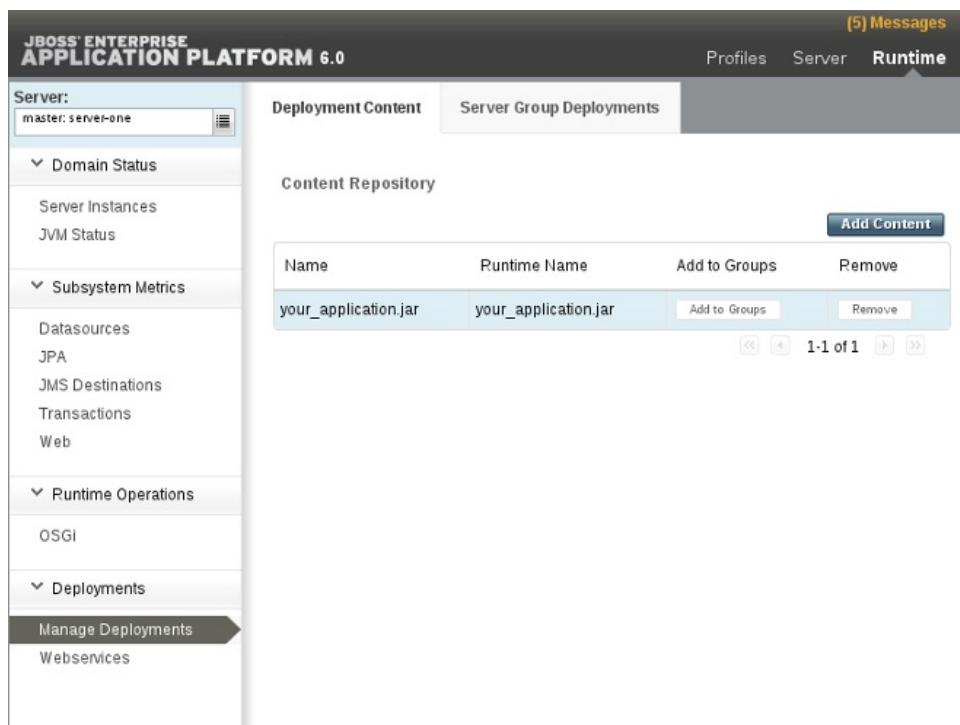


Figure 8.8. Available deployments in a managed domain

a. Disable the application in a Managed Domain

Click on the **Server Group Deployments** tab to view the server groups and the status of their deployed applications.

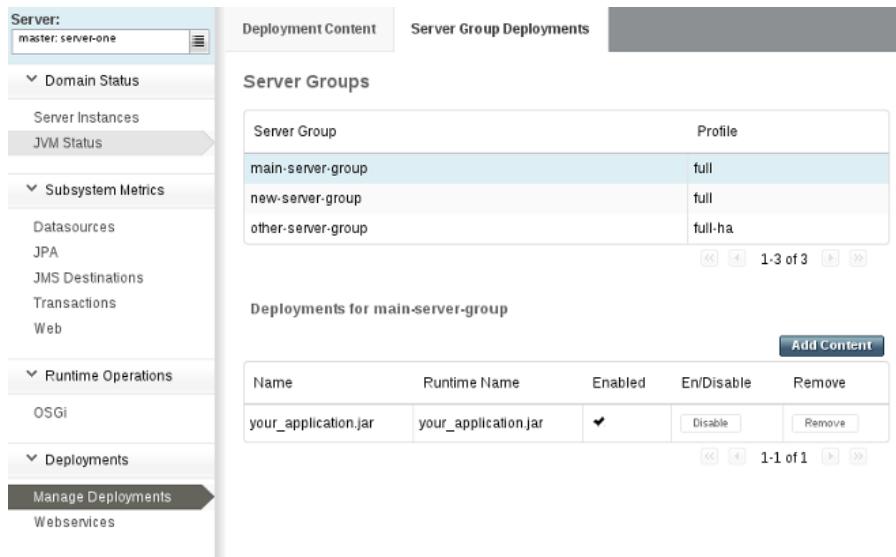


Figure 8.9. Server group deployments

b. Select server group

Click on the name of the server in the **Server Group** table to undeploy an application from.

c. Disable the application from the selected server

Click on the **disable** button to disable the application for the selected server.

d. Confirm that you wish to disable the application

Click on the **confirm** button to confirm that the application will be disabled on the server instance.

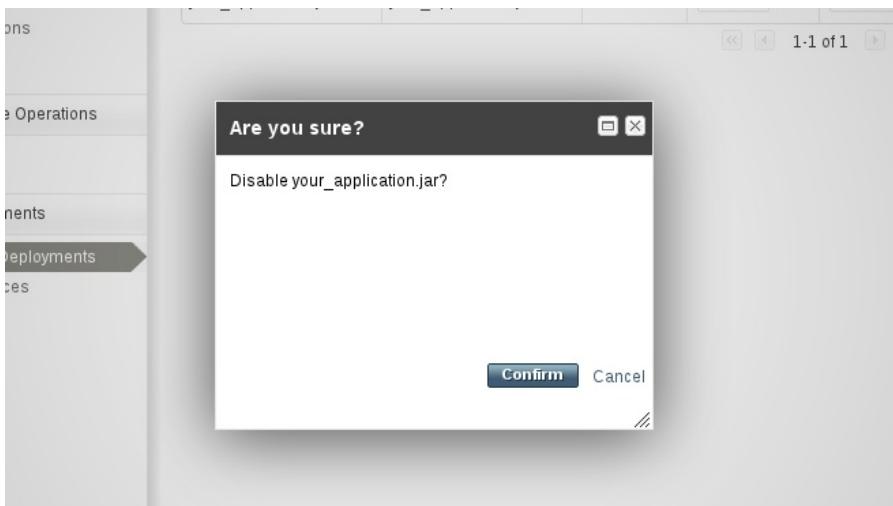


Figure 8.10. Confirm the application to disable

e. **Repeat undeployment for remaining server groups**

Repeat as required for other server groups. The application status is confirmed for each server group in the **Deployments** table.

Server Group	Profile
main-server-group	full
new-server-group	full
other-server-group	full-ha

Deployments for other-server-group				
Name	Runtime Name	Enabled	En/Disable	Remove
your_application.jar	your_application.jar	<input checked="" type="checkbox"/>	<input type="button" value="Enable"/>	<input type="button" value="Remove"/>

Figure 8.11. Confirmation of application undeployment from a server group

Result

The application is undeployed from the relevant server or server group.

[Report a bug](#)

8.3. Deploy with the Management CLI

8.3.1. Manage Application Deployment in the Management CLI

Deploying applications via the Management CLI gives you the benefit of 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 deployment status of a single server in the case of a standalone instance, or an entire network of servers in the case of a managed domain.

[Report a bug](#)

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

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)
- ▶ [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 8.3. Task

▶ Run the deploy command

From the Management CLI, enter the **deploy** command with the path to the application deployment. Include the **--all-server-groups** parameter to deploy to all server groups.

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --all-server-
groups
'test-application.war' deployed successfully.
```

- A. Alternatively, define specific server groups for the deployment with the **--server-groups** parameter.

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --server-
groups server_group_1, server_group_2
'test-application.war' deployed successfully.
```

Result

The specified application is now deployed to a server group in your managed domain.

[Report a bug](#)

8.3.3. Undeploy an Application in a Managed Domain Using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)
- ▶ [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#)
- ▶ [Section 8.3.2, "Deploy an Application in a Managed Domain Using the Management CLI"](#)

Procedure 8.4. Task

▶ Run the undeploy command

From the Management CLI, enter the **undeploy** command with the filename of the application deployment. The application can be undeployed from any server group that it was originally deployed to with the addition of the **--all-relevant-server-groups** parameter.

```
[domain@localhost:9999 /] undeploy test-application.war --all-relevant-
server-groups
Successfully undeployed test-application.war.
```

Result

The specified application is now undeployed.

[Report a bug](#)

8.3.4. Deploy an Application in a Standalone Server Using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, "Launch the Management CLI"](#)
- ▶ [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 8.5. Task

▶ Run the deploy command

From the Management CLI, enter the **deploy** command with the path to the application deployment.

```
[standalone@localhost:9999 /] deploy ~/path/to/test-application.war
'test-application.war' deployed successfully.
```

Result

The specified application is now deployed in the standalone server.

[Report a bug](#)

8.3.5. Undeploy an Application in a Standalone Server Using the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)
- ▶ [Section 3.3.4, “Connect to a Managed Server Instance Using the Management CLI”](#)
- ▶ [Section 8.3.4, “Deploy an Application in a Standalone Server Using the Management CLI”](#)

Procedure 8.6. Task

Run the undeploy command

From the Management CLI, enter the **undeploy** command with the filename of the application deployment.

```
[standalone@localhost:9999 /] undeploy test-application.war
Successfully undeployed test-application.war.
```

Result

The specified application is now undeployed.

[Report a bug](#)

8.4. Deploy with the Deployment Scanner

8.4.1. Manage Application Deployment in the Deployment Scanner

Deploying applications to a standalone server instance via the deployment scanner allows you to build and test applications in a manner suited for rapid development cycles. You can configure the deployment scanner to suit your needs for deployment frequency and behaviour for a variety of application types.

[Report a bug](#)

8.4.2. Deploy an Application to a Standalone Server Instance with the Deployment Scanner

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#)

Summary

This task shows a method for deploying applications to a standalone server instance with the deployment scanner. As indicated in the [Section 8.1, “About Application Deployment”](#) topic, this method is retained for the convenience of developers, where the Management Console and Management CLI methods are recommended for application management under production environments.

Procedure 8.7. Task

1. Copy content to the deployment folder

Copy the application file to the deployment folder found at **EAP_HOME/standalone/deployments/**.

2. Deployment scanning modes

The application deployment varies between automatic and manual deployment scanner modes.

A. Automatic deployment

The deployment scanner picks up a change to the state of the folder and creates a marker file as defined in the [Section 8.4.5, “Reference for Deployment Scanner Marker Files”](#) topic.

B. Manual deployment

The deployment scanner requires a marker file to trigger the deployment process. The following example uses the Unix **touch** command to create a new **.dodeploy** file.

Example 8.1. Deploy with the touch command

```
[user@host bin]$ touch
$EAP_HOME/standalone/deployments/example.war.dodeploy
```

Result

The application file is deployed to the application server. A marker file is created in the deployment folder to indicate the successful deployment, and the application is flagged as **Enabled** in the Management Console.

Example 8.2. Deployment folder contents after deployment

```
example.war
example.war.deployed
```

[Report a bug](#)

8.4.3. Undeploy an Application to a Standalone Server Instance with the Deployment Scanner

Prerequisites

- » [Section 2.6.1. "Start JBoss Enterprise Application Platform 6"](#)
- » [Section 8.4.2. "Deploy an Application to a Standalone Server Instance with the Deployment Scanner"](#)

Summary

This task shows a method for undeploying applications from a standalone server instance that have been deployed with the deployment scanner. As indicated in the [Section 8.1. "About Application Deployment"](#) topic, this method is retained for the convenience of developers, where the Management Console and Management CLI methods are recommended for application management under production environments.

**Note**

The deployment scanner should not be used in conjunction with other deployment methods for application management. Applications removed from the application server by the management console will be removed from the runtime without affecting the marker files or application contained in the deployment directory. To minimize the risk of accidental redeployment or other errors, use the Management CLI and Management Console for administration in production environments.

Procedure 8.8. Task

- » **Undeploy the application**

There are two methods to undeploy the application depending on whether you want to delete the application from the deployment folder or only alter its deployment status.

A. **Undeploy by deleting the marker file**

Delete the deployed application's **example.war.deployed** marker file to trigger the deployment scanner to begin undeploying the application from the runtime.

Result

The deployment scanner undeploys the application and creates a **example.war.undeployed** marker file. The application remains in the deployment folder.

B. **Undeploy by removing the application**

Remove the application from the deployment directory to trigger the deployment scanner to begin undeploying the application from the runtime.

Result

The deployment scanner undeploys the application and creates a

filename.filetype.undeployed marker file. The application is not present in the deployment folder.

Result

The application file is undeployed from the application server and is not visible in the **Deployments** screen of the Management Console.

[Report a bug](#)

8.4.4. Redeploy an Application to a Standalone Server Instance with the Deployment Scanner

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#)
- ▶ [Section 8.4.2, “Deploy an Application to a Standalone Server Instance with the Deployment Scanner”](#)

Summary

This task shows a method for redeploying applications to a standalone server instance that have been deployed with the deployment scanner. As indicated in the [Section 8.1, “About Application Deployment”](#) topic, this method is retained for the convenience of developers, where the Management Console and Management CLI methods are recommended for application management under production environments.

Procedure 8.9. Task

▶ Redeploy the application

There are three possible methods to redeploy an application deployed with the deployment scanner. These methods trigger the deployment scanner to initiate a deployment cycle, and can be chosen to suit personal preference.

A. Redeploy by altering the marker file

Trigger the deployment scanner redeployment by altering the marker file's access and modification timestamp. In the following Linux example, a Unix **touch** command is used.

Example 8.3. Redeploy with the Unix touch command

```
[user@host bin]$ touch
EAP_HOME/standalone/deployments/example.war.dodeploy
```

Result

The deployment scanner detects a change in the marker file and redeploys the application. A new **.deployed** file marker replaces the previous.

B. Redeploy by creating a new .dodeploy marker file

Trigger the deployment scanner redeployment by creating a new **.dodeploy** marker file. Refer to the manual deployment instructions in [Section 8.4.2, “Deploy an Application to a Standalone Server Instance with the Deployment Scanner”](#).

C. Redeploy by deleting the marker file

As described in [Section 8.4.5, “Reference for Deployment Scanner Marker Files”](#), deleting a deployed application's **.deployed** marker file will trigger an undeployment and create an **.undeployed** marker. Deleting the undeployment marker will trigger the deployment cycle again. Refer to [Section 8.4.3, “Undeploy an Application to a Standalone Server Instance with the Deployment Scanner”](#) for further information.

Result

The application file is redeployed.

[Report a bug](#)

8.4.5. Reference for Deployment Scanner Marker Files

Marker files are a part of the deployment scanner subsystem. These files mark the status of an application within the deployment directory of the standalone server instance. A marker file has the same name as the application, with the file suffix indicating the state of the application's deployment. The following table defines the types and responses for each marker file.

Example 8.4. Marker file example

The following example shows the marker file for a successfully deployed instance of an application called **testapplication.war**.

testapplication.war.deployed

Table 8.1. Marker filetype definitions

Filename Suffix	Origin	Description
.dodeploy	User generated	Indicates that the content should be deployed or redeployed into the runtime.
.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 pushing live. Can be used with zipped content, although the scanner detects in-progress changes to zipped content and waits until completion.
.isdeploying	System generated	Indicates the initiation of deployment. The marker file will be deleted when the deployment process completes.
.deployed	System generated	Indicates that the content has been deployed. The content will be undeployed if this file is deleted.
.failed	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 visible to the auto-deployment again.
.isundeploying	System generated	Indicates a response to a .deployed file deletion. The content will be undeployed and the marker will be automatically deleted upon completion.
.undeployed	System generated	Indicates that the content has been undeployed. Deletion of the marker file has no impact to content redeployment.
.pending	System generated	Indicates that deployment instructions will be sent to the server pending resolution of a detected issue. This marker serves as a global deployment road-block. The scanner will not instruct the server to deploy or undeploy any other content while this condition exists.

[Report a bug](#)

8.4.6. Reference for Deployment Scanner Attributes

The deployment scanner contains the following attributes that are exposed to the Management CLI and able to be configured using the **write-attribute** operation. For more information on configuration options, refer to the topic [Section 8.4.8, "Configure the Deployment Scanner with the Management CLI"](#).

Table 8.2. Deployment Scanner Attributes

Name	Description	Type	default Value
auto-deploy-expoded	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.	Boolean	False
auto-deploy-xml	Allows the automatic deployment of XML content without requiring a .dodeploy marker file.	Boolean	True
auto-deploy-zipped	Allows the automatic deployment of zipped content without requiring a .dodeploy marker file.	Boolean	True
deployment-timeout	The time value in seconds for the deployment scanner to allow a deployment attempt before being cancelled.	Long	60
path	Defines the actual filesystem path to be scanned. If the relative-to attribute is specified, the path value acts as a relative addition to that directory or path.	String	Deployment s
relative-to	Reference to a filesystem path defined in the paths section of the server configuration XML file.	String	jboss.serve r.base.dir
scan-enabled	Allows the automatic scanning for applications by scan-interval and at startup.	Boolean	True
scan-interval	The time interval in milliseconds between scans of the repository. A value of less than 1 restricts the scanner to operate only at startup.	Int	5000

[Report a bug](#)

8.4.7. Configure the Deployment Scanner

The deployment scanner can be configured using the Management Console or the Management CLI. You can create a new deployment scanner or manage the existing scanner attributes. These include the scanning interval, the location of the deployment folder, and the application file types that will trigger a deployment.

[Report a bug](#)

8.4.8. Configure the Deployment Scanner with the Management CLI

Prerequisites

- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Summary

While there are multiple methods of configuring the deployment scanner, the Management CLI can be used to expose and modify the attributes by use of batch scripts or in real time. You can modify the behaviour of the deployment scanner by use of the **read-attribute** and **write-attribute** global command line operations. Further information about the deployment scanner attributes are defined in the topic [Section 8.4.6, “Reference for Deployment Scanner Attributes”](#).

The deployment scanner is a subsystem of JBoss Enterprise Application Platform 6, and can be viewed in the **standalone.xml**.

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
  <deployment-scanner path="deployments" relative-to="jboss.server.base.dir"
  scan-interval="5000"/>
</subsystem>
```

Procedure 8.10. Task

1. Determine the deployment scanner attributes to configure

Configuring the deployment scanner via the Management CLI requires that you first expose the correct attribute names. You can do this with the **read-resources** operation at either the root node, or by using the **cd** command to change into the subsystem child node. You can also display the attributes with the **ls** command at this level.

A. Expose the deployment scanner attributes with the **read-resource** operation

Use the **read-resource** operation to expose the attributes defined by the default deployment scanner resource.

```
[standalone@localhost:9999 /] /subsystem=deployment-
scanner/scanner=default:read-resource
{
  "outcome" => "success",
  "result" => {
    "auto-deploy-expoded" => false,
    "auto-deploy-xml" => true,
    "auto-deploy-zipped" => true,
    "deployment-timeout" => 60,
    "path" => "deployments",
    "relative-to" => "jboss.server.base.dir",
    "scan-enabled" => true,
    "scan-interval" => 5000
  }
}
```

B. Expose the deployment scanner attributes with the **ls** command

Use the **ls** command with the **-l** optional argument to display a table of results that include the subsystem node attributes, values, and type. You can learn more about the **ls** command and its arguments by exposing the CLI help entry by typing **ls --help**. For more information about the help menu in the Management CLI, refer to the topic [Section 3.3.5, “Get Help with the Management CLI”](#).

```
[standalone@localhost:9999 /] ls -l /subsystem=deployment-
scanner/scanner=default
ATTRIBUTE          VALUE           TYPE
auto-deploy-expoded  false        BOOLEAN
auto-deploy-xml      true        BOOLEAN
auto-deploy-zipped   true        BOOLEAN
deployment-timeout   60          LONG
path                deployments  STRING
relative-to          jboss.server.base.dir  STRING
scan-enabled         true        BOOLEAN
scan-interval        5000        INT
```

2. Configure the deployment scanner with the **write-attribute** operation

Once you have determined the name of the attribute to modify, use the **write-attribute** to specify the attribute name and the new value to write to it. The following examples are all run at the child node level, which can be accessed by using the **cd** command and tab completion to expose and change into the default scanner node.

```
[standalone@localhost:9999 /] cd subsystem=deployment-scanner/scanner=default
```

a. Enable automatic deployment of exploded content

Use the **write-attribute** operation to enable the automatic deployment of exploded application content.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-
deploy-expoded,value=true)
{"outcome" => "success"}
```

b. Disable the automatic deployment of XML content

Use the **write-attribute** operation to disable the automatic deployment of XML application content.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-xml,value=false)
{"outcome" => "success"}
```

c. Disable the automatic deployment of zipped content

Use the **write-attribute** command to disable the automatic deployment of zipped application content.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-zipped,value=false)
{"outcome" => "success"}
```

d. Configure the path attribute

Use the **write-attribute** operation to modify the path attribute, substituting the example **newpathname** value for the new path name for the deployment scanner to monitor. Note that the server will require a reload to take effect.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=path,value=newpathname)
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

e. Configure the relative path attribute

Use the **write-attribute** operation to modify the relative reference to the filesystem path defined in the paths section of the configuration XML file. Note that the server will require a reload to take effect.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=relative-to,value=new.relative.dir)
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

f. Disable the deployment scanner

Use the **write-attribute** operation to disable the deployment scanner by setting the **scan-enabled** value to false.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}
```

g. Change the scan interval

Use the **write-attribute** operation to modify the scan interval time from 5000 milliseconds to 10000 milliseconds.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-interval,value=10000)
{"outcome" => "success"}
```

Result

Your configuration changes are saved to the deployment scanner.

[Report a bug](#)

8.5. Deploy with Maven

8.5.1. Manage Application Deployment with Maven

Deploying applications via Maven allows you to incorporate a deployment cycle as part of your existing development workflow.

[Report a bug](#)

8.5.2. Deploy an Application with Maven

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#)

Summary

This task shows a method for deploying applications with Maven. The example provided uses the **jboss-as-helloworld.war** application found in the JBoss Enterprise Application Platform 6 Quick Starts collection. The **helloworld** project contains a POM file which initializes the **jboss-as-maven-plugin**. This plugin provides simple operations to deploy and undeploy applications to and from the application server.

Procedure 8.11. Deploy an application with Maven

1. Run the Maven deploy command in a terminal session

Open a terminal session and navigate to the directory containing the quickstart examples.

2. Run the Maven deploy command to deploy the application. If the application is already running, it will be redeployed.

```
[localhost]$ mvn package jboss-as:deploy
```

3. Confirm the application deployment

A. View result in terminal window

The deployment can be confirmed by viewing the operation logs in the terminal window.

Example 8.5. Maven confirmation for helloworld application

```
[INFO] -----  
-----  
[INFO] BUILD SUCCESSFUL  
[INFO] -----  
-----  
[INFO] Total time: 3 seconds  
[INFO] Finished at: Mon Oct 10 17:22:05 EST 2011  
[INFO] Final Memory: 21M/343M  
[INFO] -----  
-----
```

B. View results in server terminal window

The deployment can also be confirmed in the status stream of the active application server instance.

Example 8.6. Application server confirmation for helloworld application

```

17:22:04,922 INFO [org.jboss.as.server.deployment] (pool-1-thread-3)
Content added at location
/home/username/EAP_Home/standalone/data/content/2c/39607b0c8dbc6a36585f7
2866c1bcfc951f3ff/content
17:22:04,924 INFO [org.jboss.as.server.deployment] (MSC service thread
1-1) Starting deployment of "jboss-as-helloworld.war"
17:22:04,954 INFO [org.jboss.weld] (MSC service thread 1-3) Processing
CDI deployment: jboss-as-helloworld.war
17:22:04,973 INFO [org.jboss.weld] (MSC service thread 1-2) Starting
Services for CDI deployment: jboss-as-helloworld.war
17:22:04,979 INFO [org.jboss.weld] (MSC service thread 1-4) Starting
weld service
17:22:05,051 INFO [org.jboss.web] (MSC service thread 1-2) registering
web context: /jboss-as-helloworld
17:22:05,064 INFO [org.jboss.as.server.controller] (pool-1-thread-3)
Deployed "jboss-as-helloworld.war"

```

Result

The application is deployed to the application server.

[Report a bug](#)

8.5.3. Undeploy an Application with Maven**Prerequisites**

- » [Section 2.6.1. "Start JBoss Enterprise Application Platform 6"](#)

Summary

This task shows a method for undeploying applications with Maven. The example provided uses the **jboss-as-helloworld.war** application found in the Enterprise Application Server Quick Starts collection. The **helloworld** project contains a POM file which initializes the **jboss-as-maven-plugin**. This plug-in provides simple operations to deploy and undeploy applications to and from the application server.

Procedure 8.12. Undeploy an application with Maven**1. Run the Maven deploy command in a terminal session**

Open a terminal session and navigate to the directory containing the quickstart examples.

Example 8.7. Change into the helloworld application directory

```
[localhost]$ cd ~/EAP_Quickstarts/helloworld
```

2. Run the Maven undeploy command to undeploy the application.

```
[localhost]$ mvn jboss-as:undeploy
```

3. Confirm the application undeployment**A. View result in terminal window**

The undeployment can be confirmed by viewing the operation logs in the terminal window.

Example 8.8. Maven confirmation for helloworld application

```
[INFO] -----
[INFO] Building JBoss AS Quickstarts: Helloworld
[INFO]   task-segment: [jboss-as:undeploy]
[INFO] -----
[INFO] [jboss-as:undeploy {execution: default-cli}]
[INFO] Executing goal undeploy for
/home/username/EAP_Quickstarts/helloworld/target/jboss-as-helloworld.war
on server localhost (127.0.0.1) port 9999.
Oct 10, 2011 5:33:02 PM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 3.2.0.Beta2
Oct 10, 2011 5:33:02 PM org.xnio.Xnio <clinit>
INFO: XNIO Version 3.0.0.Beta2
Oct 10, 2011 5:33:02 PM org.xnio.nio.NioXnio <clinit>
INFO: XNIO NIO Implementation Version 3.0.0.Beta2
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 1 second
[INFO] Finished at: Mon Oct 10 17:33:02 EST 2011
[INFO] Final Memory: 11M/212M
[INFO] -----
```

B. View results in server terminal window

The undeployment can also be confirmed in the status stream of the active application server instance.

Example 8.9. Application server confirmation for helloworld application

```
17:33:02,334 INFO  [org.jboss.weld] (MSC service thread 1-3) Stopping
weld service
17:33:02,342 INFO  [org.jboss.as.server.deployment] (MSC service thread
1-3) Stopped deployment jboss-as-helloworld.war in 15ms
17:33:02,352 INFO  [org.jboss.as.server.controller] (pool-1-thread-5)
Undeployed "jboss-as-helloworld.war"
```

Result

The application is undeployed from the application server.

[Report a bug](#)

Chapter 9. Securing JBoss Enterprise Application Platform

9.1. About the Security Subsystem

The security subsystem provides the infrastructure for all security functionality in the JBoss Enterprise Application Platform. Most configuration elements rarely need to be changed. The only configuration element which may need to be changed is whether to use *deep-copy-subject-mode*. In addition, you can configure system-wide security properties. Most of the configuration relates to *security domains*.

Deep Copy Mode

If deep copy subject mode is disabled (the default), copying a security data structure makes a reference to the original, rather than copying the entire data structure. This behavior is more efficient, but is prone to data corruption if multiple threads with the same identity clear the subject by means of a flush or logout operation.

Deep copy subject mode causes a complete copy of the data structure and all its associated data to be made, as long as they are marked cloneable. This is more thread-safe, but less efficient.

System-Wide Security Properties

You can set system-wide security properties, which are applied to `java.security.Security class`.

A *security domain* is a set of *Java Authentication and Authorization Service (JAAS)* declarative security configurations which one or more applications use to control authentication, authorization, auditing, and mapping. Three security domains are included by default: `jboss-ejb-policy`, `jboss-web-policy`, and `other`. The Management API, Management Console, and Management CLI use the `other` security domain. You can create as many security domains as you need to accomodate the needs of your applications.

[Report a bug](#)

9.2. About the Structure of the Security Subsystem

The security subsystem is configured in the managed domain or standalone configuration file. Most of the configuration elements can be configured using the web-based management console or the console-based management CLI. The following is the XML representing an example security subsystem.

Example 9.1. Example Security Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:security:1.1">
    <security-management>
        ...
    </security-management>
    <subject-factory>
        ...
    </subject-factory>
    <security-domains>
        <security-domain name="other" cache-type="default">
            <authentication>
                <login-module code="Remoting" flag="optional">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
                <login-module code="RealmUsersRoles" flag="required">
                    <module-option name="usersProperties"
value="${jboss.domain.config.dir}/application-users.properties"/>
                    <module-option name="rolesProperties"
value="${jboss.domain.config.dir}/application-roles.properties"/>
                    <module-option name="realm" value="ApplicationRealm"/>
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
            </authentication>
        </security-domain>
        <security-domain name="jboss-web-policy" cache-type="default">
            <authorization>
                <policy-module code="Delegating" flag="required"/>
            </authorization>
        </security-domain>
        <security-domain name="jboss-ejb-policy" cache-type="default">
            <authorization>
                <policy-module code="Delegating" flag="required"/>
            </authorization>
        </security-domain>
    </security-domains>
    <security-properties>
        ...
    </security-properties>
</subsystem>
```

The `<security-management>`, `<subject-factory>`, and `<security-properties>` elements are empty in the default configuration.

Each top-level element within the security subsystem contains information about a different aspect of the security configuration.

`<security-management>`

This section overrides high-level behaviors of the security subsystem. Each setting is optional. It is unusual to change any of these settings except for deep copy subject mode.

Option	Description
deep-copy-subject-mode	Specifies whether to copy or link to security tokens, for additional thread safety.
authentication-manager-class-name	Specifies an alternate AuthenticationManager implementation class name to use.
default-callback-handler-class-name	Specifies a global class name for the CallbackHandler implementation to be used with login modules.
authorization-manager-class-name	Specifies an alternate AuthorizationManager implementation class name to use.
audit-manager-class-name	Specifies an alternate AuditManager implementation class name to use.
identity-trust-manager-class-name	Specifies an alternate IdentityTrustManager implementation class name to use.

mapping-manager-class-name	Specifies the MappingManager implementation class name to use.
----------------------------	--

<subject-factory>

The subject factory controls creation of subject instances. It may use the authentication manager to verify the caller. The main use of the subject factory is for JCA components to establish a subject. It is unusual to need to modify the subject factory.

<security-domains>

A container element which holds multiple security domains. A security domain may contain information about authentication, authorization, mapping, and auditing modules, as well as JASPI authentication and JSSE configuration. Your application would specify a security domain to manage its security information.

<security-properties>

Contains names and values of properties which are set on the `java.security.Security` class.

[Report a bug](#)

9.3. Configure the Security Subsystem

The top-level configuration of the `security` subsystem includes one attribute, `deep-copy-subject-mode` which includes child elements `security-domains` and `security-properties`. You can configure the security subsystem using the Management CLI or web-based Management Console.

Deep Copy Mode

If deep copy subject mode is disabled (the default), copying a security data structure makes a reference to the original, rather than copying the entire data structure. This behavior is more efficient, but is prone to data corruption if multiple threads with the same identity clear the subject by means of a flush or logout operation.

Deep copy subject mode causes a complete copy of the data structure and all its associated data to be made, as long as they are marked cloneable. This is more thread-safe, but less efficient.

System-Wide Security Properties

You can set system-wide security properties, which are applied to class `java.security.Security` class.

Security Domains

A security domain is a set of *Java Authentication and Authorization Service (JAAS)* declarative security configurations which one or more applications use to control authentication, authorization, security auditing, and security mapping. Three security domains are included by default: `jboss-ejb-policy`, `jboss-web-policy`, and `other`. The Management API, Management Console, and Management CLI use the `other` security domain. You can create as many security domains as you need to accommodate the needs of your applications.

[Report a bug](#)

9.4. About Deep Copy Subject Mode

If `deep-copy-subject-mode` is disabled (the default), copying a security data structure makes a reference to the original, rather than copying the entire data structure. This behavior is more efficient, but is prone to data corruption if multiple threads with the same identity clear the subject by means of a flush or logout operation.

Deep copy subject mode causes a complete copy of the data structure and all its associated data to be made, as long as they are marked cloneable. This is more thread-safe, but less efficient.

Deep copy subject mode is configured as part of the security subsystem.

[Report a bug](#)

9.5. Enable Deep Copy Subject Mode

You can enable deep copy security mode from the web-based management console or the management CLI.

Procedure 9.1. Enable Deep Copy Security Mode from the Management Console

- 1. Log into the Management Console.**

The management console is usually available at a URL such as <http://127.0.0.1:9990/>. Adjust this value to suit your needs.

- 2. Managed Domain: Select the appropriate profile.**

In a managed domain, the security subsystem is configured per profile, and you can enable or disable the deep copy security mode in each, independently.

To select a profile, click the **Profiles** label at the top right of the console display, and then select the profile you wish to change from the **Profile** selection box at the top left.

- 3. Open the Security Subsystem configuration menu.**

Expand the **Security** menu item at the right of the management console, then click the **Security Subsystem** link.

- 4. Modify the deep-copy-subject-mode value.**

Click the **Edit** button. Check the box beside **Deep Copy Subjects:** to enable deep copy subject mode.

Enable Deep Copy Subject Mode Using the Management CLI

If you prefer to use the management CLI to enable this option, use one of the following commands.

Example 9.2. Managed Domain

```
/profile=full/subsystem=security:write-attribute(name=deep-copy-subject-mode,value=TRUE)
```

Example 9.3. Standalone Server

```
/subsystem=security:write-attribute(name=deep-copy-subject-mode,value=TRUE)
```

[Report a bug](#)

9.6. Security Domains

9.6.1. About Security Domains

Security domains are part of the JBoss Enterprise Application Platform security subsystem. All security configuration is now managed centrally, by the domain controller of a managed domain, or by the standalone server.

A security domain consists of configurations for authentication, authorization, security mapping, and auditing. It implements *Java Authentication and Authorization Service (JAAS)* declarative security.

Authentication refers to verifying the identity of a user. In security terminology, this user is referred to as a *principal*. Although authentication and authorization are different, many of the included authentication modules also handle authorization.

An *authorization* is a security policy, which contains information about actions which are allowed or prohibited. In security terminology, this is often referred to as a role.

Security mapping refers to the ability to add, modify, or delete information from a principal, role, or attribute before passing the information to your application.

The auditing manager allows you to configure *provider modules* to control the way that security events are reported.

If you use security domains, you can remove all specific security configuration from your application

itself. This allows you to change security parameters centrally. One common scenario that benefits from this type of configuration structure is the process of moving applications between testing and production environments.

[Report a bug](#)

9.6.2. About Picketbox

Picketbox is the foundational security framework that provides the authentication, authorization, audit and mapping capabilities to Java applications running in the JBoss Enterprise Application Platform. It provides the following capabilities, in a single framework with a single configuration:

- » [Section 9.6.3, "About Authentication"](#)
- » [Section 9.6.5, "About Authorization"](#) and access control
- » [Section 9.6.7, "About Security Auditing"](#)
- » [Section 9.6.9, "About Security Mapping"](#) of principals, roles, and attributes

The Picketbox configuration uses a mark-up language called *eXtensible Access Control Markup Language (XACML)*.

[Report a bug](#)

9.6.3. About Authentication

Authentication refers to identifying a subject and verifying the authenticity of the identification. The most common authentication mechanism is a username and password combination. Other common authentication mechanisms use shared keys, smart cards, or fingerprints. The outcome of a successful authentication is referred to as a principal, in terms of Java Enterprise Edition declarative security.

The JBoss Enterprise Application Platform uses a pluggable system of authentication modules to provide flexibility and integration with the authentication systems you already use in your organization. Each security domain contains one or more configured authentication modules. Each module includes additional configuration parameters to customize its behavior. The easiest way to configure the authentication subsystem is within the web-based management console.

Authentication is not the same as authorization, although they are often linked. Many of the included authentication modules can also handle authorization.

[Report a bug](#)

9.6.4. Configure Authentication in a Security Domain

To configure authentication settings for a security domain, log into the management console and follow this procedure.

Procedure 9.2. Task

1. Open the security domain's detailed view.

Click the **Profiles** label at the top right of the management console. In a managed domain, select the profile to modify from the **Profile** selection box at the top left of the Profile view. Click the **Security** menu item at the left, and click **Security Domains** from the expanded menu. Click the **View** link for the security domain you want to edit.

2. Navigate to the Authentication subsystem configuration.

Click the **Authentication** label at the top of the view if it is not already selected.

The configuration area is divided into two areas: **Login Modules** and **Details**. The login module is the basic unit of configuration. A security domain can include several login modules, each of which can include several attributes and options.

3. Add an authentication module.

Click the **Add** button to add a JAAS authentication module. Fill in the details for your module. The **Code** is the class name of the module. The **Flags** controls how the module relates to other authentication modules within the same security domain.

Explanation of the Flags

The Java Enterprise Edition 6 specification provides the following explanation of the flags for security modules. The following list is taken from

<http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Appendix>

A. Refer to that document for more detailed information.

Flag	Details
Required	The LoginModule is required to succeed. If it succeeds or fails, authentication still continues to proceed down the LoginModule list.
Requisite	LoginModule is required to succeed. If it succeeds, authentication continues down the LoginModule list. If it fails, control immediately returns to the application (authentication does not proceed down the LoginModule list).
Sufficient	The LoginModule is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the LoginModule list). If it fails, authentication continues down the LoginModule list.
Optional	The LoginModule is not required to succeed. If it succeeds or fails, authentication still continues to proceed down the LoginModule list.

After you have added your module, you can modify its **Code** or **Flags** by clicking the **Edit** button in the **Details** section of the screen. Be sure the **Attributes** tab is selected.

4. Optional: Add, edit, or remove module options.

If you need to add options to your module, click its entry in the **Login Modules** list, and select the **Module Options** tab in the **Details** section of the page. Click the **Add** button, and provide the key and value for the option. To edit an option that already exists, click the key or to change it. Use the **Remove** button to remove an option.

Result

Your authentication module is added to the security domain, and is immediately available to applications which use the security domain.

The `jboss.security.security_domain` Module Option

By default, each login module defined in a security domain has the `jboss.security.security_domain` module option added to it automatically. This option causes problems with login modules which check to make sure that only known options are defined. The IBM Kerberos login module, `com.ibm.security.auth.module.Krb5LoginModule` is one of these.

You can disable the behavior of adding this module option by setting the system property to `true` when starting JBoss Enterprise Application Platform. Add the following to your start-up parameters.

```
-Djboss.security.disable.secdomain.option=true
```

You can also set this property using the web-based Management Console. In a standalone server, you can set system properties in the **Profile** section of the configuration. In a managed domain, you can set system properties for each server group.

[Report a bug](#)

9.6.5. About Authorization

Authorization is a mechanism for granting or denying access to a resource based on identity. It is implemented as a set of declarative security roles which can be granted to principals.

The JBoss Enterprise Application Platform uses a modular system to configure authorization. Each security domain can contain one or more authorization policies. Each policy has a basic module which defines its behavior. It is configured through specific flags and attributes. The easiest way to configure the authorization subsystem is by using the web-based management console.

Authorization is different from authentication, and usually happens after authentication. Many of the authentication modules also handle authorization.

[Report a bug](#)

9.6.6. Configure Authorization in a Security Domain

To configure authorization settings for a security domain, log into the management console and follow this procedure.

Procedure 9.3. Task

1. Open the security domain's detailed view.

Click the **Profiles** label at the top right of the management console. In a managed domain, select the profile to modify from the **Profile** selection box at the top left of the Profile view. Click the **Security** menu item at the left, and click **Security Domains** from the expanded menu. Click the **View** link for the security domain you want to edit.

2. Navigate to the Authorization subsystem configuration.

Click the **Authorization** label at the top of the view if it is not already selected.

The configuration area is divided into two areas: **Policies** and **Details**. The login module is the basic unit of configuration. A security domain can include several authorization policies, each of which can include several attributes and options.

3. Add a policy.

Click the **Add** button to add a JAAS authorization policy module. Fill in the details for your module. The **Code** is the class name of the module. The **Flags** controls how the module relates to other authorization policy modules within the same security domain.

Explanation of the Flags

The Java Enterprise Edition 6 specification provides the following explanation of the flags for security modules. The following list is taken from

http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Appendix_A. Refer to that document for more detailed information.

Flag	Details
Required	The LoginModule is required to succeed. If it succeeds or fails, authorization still continues to proceed down the LoginModule list.
Requisite	LoginModule is required to succeed. If it succeeds, authorization continues down the LoginModule list. If it fails, control immediately returns to the application (authorization does not proceed down the LoginModule list).
Sufficient	The LoginModule is not required to succeed. If it does succeed, control immediately returns to the application (authorization does not proceed down the LoginModule list). If it fails, authorization continues down the LoginModule list.
Optional	The LoginModule is not required to succeed. If it succeeds or fails, authorization still continues to proceed down the LoginModule list.

After you have added your module, you can modify its **Code** or **Flags** by clicking the **Edit** button in the **Details** section of the screen. Be sure the **Attributes** tab is selected.

4. Optional: Add, edit, or remove module options.

If you need to add options to your module, click its entry in the **Login Modules** list, and select the **Module Options** tab in the **Details** section of the page. Click the **Add** button, and provide the key and value for the option. To edit an option that already exists, click the key or to change it. Use the **Remove** button to remove an option.

Result

Your authorization policy module is added to the security domain, and is immediately available to applications which use the security domain.

[Report a bug](#)

9.6.7. About Security Auditing

Security auditing refers to triggering events, such as writing to a log, in response to an event that

happens within the security subsystem. Auditing mechanisms are configured as part of a security domain, along with authentication, authorization, and security mapping details.

Auditing uses *provider modules*. You can use one of the included ones, or implement your own.

[Report a bug](#)

9.6.8. Configure Security Auditing

To configure security auditing settings for a security domain, log into the management console and follow this procedure.

Procedure 9.4. Task

1. Open the security domain's detailed view.

Click the **Profiles** label at the top right of the management console. In a standalone server, the tab is labeled **Profile**. In a managed domain, select the profile to modify from the **Profile** selection box at the top left of the Profile view. Click the **Security** menu item at the left, and click **Security Domains** from the expanded menu. Click the **View** link for the security domain you want to edit.

2. Navigate to the Auditing subsystem configuration.

Click the **Audit** label at the top of the view if it is not already selected.

The configuration area is divided into two areas: **Provider Modules** and **Details**. The provider module is the basic unit of configuration. A security domain can include several provider modules each of which can include attributes and options.

3. Add a provider module.

Click the **Add** button to add a provider module. Fill in the **Code** section with the classname of the provider module.

After you have added your module, you can modify its **Code** by clicking the **Edit** button in the **Details** section of the screen. Be sure the **Attributes** tab is selected.

4. Optional: Add, edit, or remove module options.

If you need to add options to your module, click its entry in the **Modules** list, and select the **Module Options** tab in the **Details** section of the page. Click the **Add** button, and provide the key and value for the option. To edit an option that already exists, remove it by clicking the **Remove** label, and add it again with the correct options by clicking the **Add** button.

Result

Your security auditing module is added to the security domain, and is immediately available to applications which use the security domain.

[Report a bug](#)

9.6.9. About Security Mapping

Security mapping allows you to combine authentication and authorization information after the authentication or authorization happens, but before the information is passed to your application. One example of this is using an X509 certificate for authentication, and then converting the principal from the certificate to a logical name which your application can display.

You can map principals (authentication), roles (authorization), or credentials (attributes which are not principals or roles).

Role Mapping is used to add, replace, or remove roles to the subject after authentication.

Principal mapping is used to modify a principal after authentication.

Attribute mapping is used to convert attributes from an external system to be used by your application, and vice versa.

[Report a bug](#)

9.6.10. Configure Security Mapping in a Security Domain

To configure security mapping settings for a security domain, log into the management console and follow this procedure.

Procedure 9.5. Task

- 1. Open the security domain's detailed view.**

Click the **Profiles** label at the top right of the management console. This tab is labeled **Profile** in a standalone server. In a managed domain, select the profile to modify from the **Profile** selection box at the top left of the Profile view. Click the **Security** menu item at the left, and click **Security Domains** from the expanded menu. Click the **View** link for the security domain you want to edit.

- 2. Navigate to the Mapping subsystem configuration.**

Click the **Mapping** label at the top of the view if it is not already selected.

The configuration area is divided into two areas: **Modules** and **Details**. The mapping module is the basic unit of configuration. A security domain can include several mapping modules, each of which can include several attributes and options.

- 3. Add a module.**

Click the **Add** button to add a security mapping module. Fill in the details for your module. The **Code** is the class name of the module. The **Type** field refers to the type of mapping this module performs. Allowed values are principal, role, attribute or credential.

After you have added your module, you can modify its **Code** or **Type** by clicking the **Edit** button in the **Details** section of the screen. Be sure the **Attributes** tab is selected.

- 4. Optional: Add, edit, or remove module options.**

If you need to add options to your module, click its entry in the **Modules** list, and select the **Module Options** tab in the **Details** section of the page. Click the **Add** button, and provide the key and value for the option. To edit an option that already exists, click the **Remove** label key to remove it, and add it again with the new value. Use the **Remove** button to remove an option.

Result

Your security mapping module is added to the security domain, and is immediately available to applications which use the security domain.

[Report a bug](#)

9.6.11. Use a Security Domain in Your Application

Overview

To use a security domain in your application, first you must configure the domain in either the server's configuration file or the application's descriptor file. Then you must add the required annotations to the EJB that uses it. This topic covers the steps required to use a security domain in your application.

Procedure 9.6. Configure Your Application to Use a Security Domain

- 1. Define the Security Domain**

You can define the security domain either in the server's configuration file or the application's descriptor file.

- A. Configure the security domain in the server's configuration file**

The security domain is configured in the **security** subsystem of the server's configuration file. If the JBoss Enterprise Application Platform instance is running in a managed domain, this is the **domain/configuration/domain.xml** file. If the JBoss Enterprise Application Platform instance is running as a standalone server, this is the **standalone/configuration/standalone.xml** file.

The **other**, **jboss-web-policy**, and **jboss-ejb-policy** security domains are provided by default in JBoss Enterprise Application Platform 6. The following XML example was copied from the **security** subsystem in the server's configuration file.

```

<subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="other" cache-type="default">
            <authentication>
                <login-module code="Remoting" flag="optional">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
                <login-module code="RealmDirect" flag="required">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
            </authentication>
        </security-domain>
        <security-domain name="jboss-web-policy" cache-type="default">
            <authorization>
                <policy-module code="Delegating" flag="required"/>
            </authorization>
        </security-domain>
        <security-domain name="jboss-ejb-policy" cache-type="default">
            <authorization>
                <policy-module code="Delegating" flag="required"/>
            </authorization>
        </security-domain>
    </security-domains>
</subsystem>

```

You can configure additional security domains as needed using the Management Console or CLI.

B. Configure the security domain in the application's descriptor file

The security domain is specified in the `<security-domain>` child element of the `<jboss-web>` element in the application's `WEB-INF/jboss-web.xml` file. The following example configures a security domain named `my-domain`.

```

<jboss-web>
    <security-domain>my-domain</security-domain>
</jboss-web>

```

This is only one of many settings which you can specify in the `WEB-INF/jboss-web.xml` descriptor.

2. Add the Required Annotation to the EJB

You configure security in the EJB using the `@SecurityDomain` and `@RolesAllowed` annotations. The following EJB code example limits access to the `other` security domain by users in the `guest` role.

```

package example.ejb3;

import java.security.Principal;

import javax.annotation.Resource;
import javax.annotation.security.RolesAllowed;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;

import org.jboss.ejb3.annotation.SecurityDomain;

/*
 * Simple secured EJB using EJB security annotations
 * Allow access to "other" security domain by users in a "guest" role.
 */
@Stateless
@RolesAllowed({ "guest" })
@SecurityDomain("other")
public class SecuredEJB {

    // Inject the Session Context
    @Resource
    private SessionContext ctx;

    /**
     * Secured EJB method using security annotations
     */
    public String getSecurityInfo() {
        // Session context injected using the resource annotation
        Principal principal = ctx.getCallerPrincipal();
        return principal.toString();
    }
}

```

For more code examples, see the **ejb-security** quickstart in the JBoss Enterprise Application Platform 6 Quickstarts bundle, which is available from the Red Hat Customer Portal.

[Report a bug](#)

9.6.12. Java Authorization Contract for Containers (JACC)

9.6.12.1. About Java Authorization Contract for Containers (JACC)

Java Authorization Contract for Containers (JACC) is a standard which defines a contract between containers and authorization service providers, which results in the implementation of providers for use by containers. It was defined in JSR-115, which can be found on the Java Community Process website at <http://jcp.org/en/jsr/detail?id=115>. It has been part of the core Java Enterprise Edition (Java EE) specification since Java EE version 1.3.

JBoss Enterprise Application Platform implements support for JACC within the security functionality of the security subsystem.

[Report a bug](#)

9.6.12.2. Configure Java Authorization Contract for Containers (JACC) Security

To configure Java Authorization Contract for Containers (JACC), you need to configure your security domain with the correct module, and then modify your **jboss-web.xml** to include the correct parameters.

Add JACC Support to the Security Domain

To add JACC support to the security domain, add the **JACC** authorization policy to the authorization stack of the security domain, with the **required** flag set. The following is an example of a security domain with JACC support. However, the security domain is configured in the Management Console or Management CLI, rather than directly in the XML.

```
<security-domain name="jacc" cache-type="default">
  <authentication>
    <login-module code="UsersRoles" flag="required">
      </login-module>
  </authentication>
  <authorization>
    <policy-module code="JACC" flag="required"/>
  </authorization>
</security-domain>
```

Configure a Web Application to use JACC

The **jboss-web.xml** is located in the **META-INF/** or **WEB-INF/** directory of your deployment, and contains overrides and additional JBoss-specific configuration for the web container. To use your JACC-enabled security domain, you need to include the **<security-domain>** element, and also set the **<use-jboss-authorization>** element to **true**. The following application is properly configured to use the JACC security domain above.

```
<jboss-web>
  <security-domain>jacc</security-domain>
  <use-jboss-authorization>true</use-jboss-authorization>
</jboss-web>
```

Configure an EJB Application to Use JACC

Configuring EJBs to use a security domain and to use JACC differs from Web Applications. For an EJB, you can declare *method permissions* on a method or group of methods, in the **ejb-jar.xml** descriptor. Within the **<ejb-jar>** element, any child **<method-permission>** elements contain information about JACC roles. Refer to the example configuration for more details. The **EJBMethodPermission** class is part of the Java Enterprise Edition 6 API, and is documented at <http://docs.oracle.com/javaee/6/api/javax/security/jacc/EJBMethodPermission.html>.

Example 9.4. Example JACC Method Permissions in an EJB

```
<ejb-jar>
  <method-permission>
    <description>The employee and temp-employee roles may access any method of the EmployeeService bean </description>
    <role-name>employee</role-name>
    <role-name>temp-employee</role-name>
    <method>
      <ejb-name>EmployeeService</ejb-name>
      <method-name>*</method-name>
    </method>
  </method-permission>
</ejb-jar>
```

You can also constrain the authentication and authorization mechanisms for an EJB by using a security domain, just as you can do for a web application. Security domains are declared in the **jboss-ejb3.xml** descriptor, in the **<security>** child element. In addition to the security domain, you can also specify the *run-as principal*, which changes the principal the EJB runs as.

Example 9.5. Example Security Domain Declaration in an EJB

```
<security>
  <ejb-name>*</ejb-name>
  <security-domain>myDomain</s:security-domain>
  <run-as-principal>myPrincipal</s:run-as-principal>
</s:security>
```

[Report a bug](#)

9.6.13. Java Authentication SPI for Containers (JASPI)

9.6.13.1. About Java Authentication SPI for Containers (JASPI) Security

Java Application SPI for Containers (JASPI or JASPIC) is a pluggable interface for Java applications. It is defined in JSR-196 of the Java Community Process. Refer to <http://www.jcp.org/en/jsr/detail?id=196> for details about the specification.

[Report a bug](#)

9.6.13.2. Configure Java Authentication SPI for Containers (JASPI) Security

To authenticate against a JASPI provider, add a `<authentication-jaspi>` element to your security domain. The configuration is similar to a standard authentication module, but login module elements are enclosed in a `<login-module-stack>` element. The structure of the configuration is:

Example 9.6. Structure of the `authentication-jaspi` element

```
<authentication-jaspi>
  <login-module-stack name="...">
    <login-module code="..." flag="...">
      <module-option name="..." value="..."/>
    </login-module>
  </login-module-stack>
  <auth-module code="..." login-module-stack-ref="...">
    <module-option name="..." value="..."/>
  </auth-module>
</authentication-jaspi>
```

The login module itself is configured in exactly the same way as a standard authentication module.

Because the web-based management console does not expose the configuration of JASPI authentication modules, you need to stop the JBoss Enterprise Application Platform completely before adding the configuration directly to the `EAP_HOME/domain/configuration/domain.xml` or `EAP_HOME/standalone/configuration/standalone.xml`.

[Report a bug](#)

9.7. Management Interface Security

9.7.1. Default User Security Configuration

Introduction

All management interfaces in JBoss Enterprise Application Platform 6 are secured by default. This security takes two different forms:

- » Local interfaces are secured by a SASL contract between local clients and the server they connect to. This security mechanism is based on the client's ability to access the local filesystem. This is because access to the local filesystem would allow the client to add a user or otherwise change the configuration to thwart other security mechanisms. This adheres to the principle that if physical access to the filesystem is achieved, other security mechanisms are superfluous. The mechanism happens in four steps:

 **Note**

HTTP access is considered to be remote, even if you connect to the localhost using HTTP.

1. The client sends a message to the server which includes a request to authenticate with the local SASL mechanism.
2. The server generates a one-time token, writes it to a unique file, and sends a message to the client with the full path of the file.
3. The client reads the token from the file and sends it to the server, verifying that it has local access to the filesystem.

4. The server verifies the token and then deletes the file.

- ▶ Remote clients, including local HTTP clients, use realm-based security. The default realm with the permissions to configure the JBoss Enterprise Application Platform 6 remotely using the management interfaces is **ManagementRealm**. A script is provided which allows you to add users to this realm (or realms you create). For more information on adding users, refer to the Getting Started chapter of the Installation guide for JBoss Enterprise Application Platform 6. For each user, the username, a hashed password, and the realm are stored in a file. The file is in a different location if the JBoss Enterprise Application Platform 6 is configured as a managed domain or a standalone server.

Managed domain

`EAP_HOME/domain/configuration/mgmt-users.properties`

Standalone server

`EAP_HOME/standalone/configuration/mgmt-users.properties`

Even though the contents of the **mgmt-users.properties** are masked, the file should still be treated as a sensitive file. It is recommended that it be set to the file mode of **600**, which gives no access other than read and write access by the file owner.

[Report a bug](#)

9.7.2. Overview of Advanced Management Interface Configuration

The Management interface configuration in the `EAP_HOME/domain/configuration/host.xml` or `EAP_HOME/standalone/configuration/standalone.xml` controls which network interfaces the host controller process binds to, which types of management interfaces are available at all, and which type of authentication system is used to authenticate users on each interface. This topic discusses how to configure the Management Interfaces to suit your environment.

The Management subsystem consists of a `<management>` element that includes several configurable attributes, and the following three configurable child elements. The security realms and outbound connections are each first defined, and then applied to the management interfaces as attributes.

- ▶ `<security-realms>`
- ▶ `<outbound-connections>`
- ▶ `<management-interfaces>`

Security Realms

The security realm is responsible for the authentication and authorization of users allowed to administer the JBoss Enterprise Application Platform via the Management API, Management CLI, or web-based Management Console.

Two different file-based security realms are included in a default installation: **ManagementRealm** and **ApplicationRealm**. Each of these security realms uses a `-users.properties` file to store users and hashed passwords, and a `-roles.properties` to store mappings between users and roles. Support is also included for an LDAP-enabled security realm.



Note

Security realms can also be used for your own applications. The security realms discussed here are specific to the management interfaces.

Outbound Connections

Some security realms connect to external interfaces, such as an LDAP server. An outbound connection defines how to make this connection. A pre-defined connection type, **ldap-connection**, sets all of the required and optional attributes to connect to the LDAP server and verify the credential.

Management Interfaces

A management interface includes properties about how connect to and configure JBoss Enterprise

Application Platform. Such information includes the named network interface, port, security realm, and other configurable information about the interface. Two interfaces are included in a default installation:

- » **http-interface** is the configuration for the web-based Management Console.
- » **native-interface** is the configuration for the command-line Management CLI and the REST-like Management API.

Each of the three main configurable elements of the host management subsystem are interrelated. A security realm refers to an outbound connection, and a management interface refers to a security realm.

[Report a bug](#)

9.7.3. About LDAP

Lightweight Directory Access Protocol (LDAP) is a protocol for storing and distributing directory information across a network. This directory information includes information about users, hardware devices, access roles and restrictions, and other information.

Some common implementations of LDAP include OpenLDAP, Microsoft Active Directory, IBM Tivoli Directory Server, Oracle Internet Directory, and others.

JBoss Enterprise Application Platform includes several authentication and authorization modules which allow you to use a LDAP server as the authentication and authorization authority for your Web and EJB applications.

[Report a bug](#)

9.7.4. Use LDAP to Authenticate to the Management Interfaces

To use an LDAP directory server as the authentication source for the Management Console, Management CLI, or Management API, you need to perform the following procedures:

1. Create an outbound connection to the LDAP server.
2. Create an LDAP-enabled security realm.
3. Reference the new security domain in the Management Interface.

Create an Outbound Connection to an LDAP Server

The LDAP outbound connection allows the following attributes:

Table 9.1. Attributes of an LDAP Outbound Connection

Attribute	Required	Description
name	yes	The name to identify this connection. This name is used in the security realm definition.
url	yes	The URL address of the directory server.
search-dn	yes	The fully distinguished name (DN) of the user authorized to perform searches.
search-credentials	yes	The password of the user authorized to perform searches.
initial-context-factory	no	The initial context factory to use when establishing the connection. Defaults to com.sun.jndi.ldap.LdapContextFactory .

Example 9.7. Add an LDAP Outbound Connection

This example adds an outbound connection with the following properties set:

- » Search DN: **cn=search,dc=acme,dc=com**
- » Search Credential: **myPass**
- » URL: **http://127.0.0.1**

```
/host=master/core-service=management/ldap-
connection=ldap_connection/:add(search-
credential=myPass,url=http://127.0.0.1,search-dn=cn=search,dc=acme,dc=com)
```

Example 9.8. XML Representing an LDAP Outbound Connection

```
<outbound-connections>
  <ldap name="ldap_connection" url="ldap://127.0.0.1" search-
dn="cn=search,dc=acme,dc=com" search-credential="myPass" />
</outboundconnections>
```

Create an LDAP-Enabled Security Realm

The Management Interfaces can authenticate against LDAP server instead of the property-file based security realms configured by default. The LDAP authenticator operates by first establishing a connection to the remote directory server. It then performs a search using the username which the user passed to the authentication system, to find the fully-qualified distinguished name (DN) of the LDAP record. A new connection is established, using the DN of the user as the credential, and password supplied by the user. If this authentication to the LDAP server is successful, the DN is verified to be valid.

The LDAP security realm needs the following configuration attributes and elements in order to perform its functions.

connection

The name of the connection defined in **<outbound-connections>** to use to connect to the LDAP directory.

base-dn

The distinguished name of the context to begin searching for the user.

recursive

Whether the search should be recursive throughout the LDAP directory tree, or only search the specified context. Defaults to **false**.

user-dn

The attribute of the user that holds the distinguished name. This is subsequently used to test authentication as the user can complete. Defaults to **dn**.

One of username-filter or advanced-filter, as a child element

The **username-filter** takes a single attribute called **attribute**, whose value is the name of the LDAP attribute which holds the username, such as **user_name** or **sambaAccountName**.

The **advanced-filter** takes a single attribute called **filter**. This attribute contains a filter query in standard LDAP syntax. Be cautious to escape any & characters by changing them to &. An example of a filter is:

```
(&(sAMAccountName={0})(memberOf=cn=admin,cn=users,dc=acme,dc=com))
```

After escaping the ampersand character, the filter appears as:

```
(&#38;(sAMAccountName={0})(memberOf=cn=admin,cn=users,dc=acme,dc=com))
```

Example 9.9. XML Representing an LDAP-enabled Security Realm

This example uses the following parameters:

- » connection - **ldap_connection**
- » base-dn - **cn=users,dc=acme,dc=com**.
- » username-filter - **attribute="sambaAccountName"**

```
<security-realm name="TestRealm">
    <authentication>
        <ldap connection="ldap_connection" base-dn="cn=users,dc=acme,dc=com">
            <username-filter attribute="sambaAccountName" />
        </ldap>
    </authentication>
</security-realm>
```

Example 9.10. Add an LDAP Security Realm

The command below adds a security realm and sets its attributes.

```
/host=master/core-service=management/security-
realm=ldap_security_realm/authentication=ldap:add(base-dn="DC=mycompany,DC=org",
recursive=true, username-attribute="MyAccountName",
connection="ldap_connection")
```

Apply the New Security Realm to the Management Interface

After you create a security realm, you need to reference it in the configuration of your management interface. The management interface will use the security realm for HTTP digest authentication.

Example 9.11. Apply the Security Realm to the HTTP Interface

After this configuration is in place, the web-based Management Console will use LDAP to authenticate its users.

```
/host=master/core-service=management/management-interface=http-interface/:write-
attribute(name=security-realm,value=TestRealm)
```

Restart JBoss Enterprise Application Platform and your HTTP interface uses your LDAP server for authentication.

[Report a bug](#)

9.7.5. Disable the HTTP Management Interface

In a managed domain, you only need access to the HTTP interface on the domain controller, rather than on domain member servers. In addition, on a production server, you may decide to disable the web-based Management Console altogether.

 **Note**

Other clients, such as JBoss Operations Network, also operate using the HTTP interface. If you want to use these services, and simply disable the Management Console itself, you can set the **console-enabled-attribute** of the HTTP interface to **false**, instead of disabling the interface completely.

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

To disable access to the HTTP interface, which also disables access to the web-based Management Console, you can delete the HTTP interface altogether.

The following JBoss CLI command allows you to read the current contents of your HTTP interface, in case you decide to add it again.

Example 9.12. Read the Configuration of the HTTP Interface

```
/host=master/core-service=management/management-interface=http-
interface/:read-resource(recursive=true,proxies=false,include-
runtime=false,include-defaults=true)
{
    "outcome" => "success",
    "result" => {
        "console-enabled" => true,
        "interface" => "management",
        "port" => expression "${jboss.management.http.port:9990}",
        "secure-port" => undefined,
        "security-realm" => "ManagementRealm"
    }
}
```

To remove the HTTP interface, issue the following command:

Example 9.13. Remove the HTTP Interface

```
/host=master/core-service=management/management-interface=http-interface/:remove
```

To re-enable access, issue the following commands to re-create the HTTP Interface with the default values.

Example 9.14. Re-CREATE the HTTP Interface

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

```
/host=master/core-service=management/management-interface=http-interface/:write-
attribute(name=interface,value=management)
```

```
/host=master/core-service=management/management-interface=http-interface/:write-
attribute(name=port,value=${jboss.management.http.port:9990})
```

```
/host=master/core-service=management/management-interface=http-interface/:write-
attribute(name=security-realm,value=ManagementRealm)
```

[Report a bug](#)

9.7.6. Remove Silent Authentication from the Default Security Realm

Summary

The default installation of JBoss Enterprise Application Platform 6 contains a method of silent authentication for a local Management CLI user. This allows the local user the ability to access the Management CLI without username or password authentication. This functionality is enabled as a convenience, and to assist local users running Management CLI scripts without requiring authentication. It is considered a useful feature given that access to the local configuration typically also gives the user the ability to add their own user details or otherwise disable security checks.

The convenience of silent authentication for local users can be disabled where greater security control is required. This can be achieved by removing the **local** element within the **security-realm** section of the configuration file. This applies to both the **standalone.xml** for a Standalone Server instance, or **host.xml** for a Managed Domain. You should only consider the removal of the **local** element if you understand the impact that it might have on your particular server configuration.

The preferred method of removing silent authentication is by use of the Management CLI, which directly removes the **local** element visible in the following example.

Example 9.15. Example of the local element in the security-realm

```
<security-realms>
    <security-realm name="ManagementRealm">
        <authentication>
            <local default-user="$local"/>
            <properties path="mgmt-users.properties" relative-
to="jboss.server.config.dir"/>
        </authentication>
    </security-realm>
    <security-realm name="ApplicationRealm">
        <authentication>
            <local default-user="$local" allowed-users="*"/>
            <properties path="application-users.properties" relative-
to="jboss.server.config.dir"/>
        </authentication>
        <authorization>
            <properties path="application-roles.properties" relative-
to="jboss.server.config.dir"/>
        </authorization>
    </security-realm>
</security-realms>
```

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#)
- ▶ [Section 3.3.2, “Launch the Management CLI”](#)

Procedure 9.7. Task

Remove silent authentication with the Management CLI

Remove the **local** element from the Management Realm and Application Realm as required.

1. Remove the **local** element from the Management Realm.

```
/core-service=management/security-
realm=ManagementRealm/authentication=local:remove
```

2. Remove the **local** element from the Application Realm.

```
/core-service=management/security-
realm=ApplicationRealm/authentication=local:remove
```

Result

The silent authentication mode is removed from the **ManagementRealm** and the **ApplicationRealm**.

[Report a bug](#)

9.7.7. Disable Remote Access to the JMX Subsystem

Remote JMX connectivity allows you to trigger JDK and application management operations. In order to secure an installation, disable this function. You can do this either by removing the remote connection configuration, or removing the JMX subsystem entirely. The JBoss CLI commands reference the default profile in a managed domain configuration. To modify a different profile, modify the **/profile=default** part of the command. For a standalone server, remove that portion of the command completely.

Note

The remoting connector is removed from the JMX subsystem by default. This command is provided for your information, in case you add it during development.

Example 9.16. Remove the Remote Connector from the JMX Subsystem

```
/profile=default/subsystem=jmx/remoting-connector=jmx/:remove
```

Example 9.17. Remove the JMX Subsystem

Run this command for each profile you use, if you use a managed domain.

```
/profile=default/subsystem=jmx/:remove
```

[Report a bug](#)

9.7.8. Configure Security Realms for the Management Interfaces

The Management Interfaces use security realms to control authentication and access to the configuration mechanisms of JBoss Enterprise Application Platform. This topic shows you how to read and configure security realms. These commands use the Management CLI.

Read a Security Realm's Configuration

This example shows the default configuration for the **ManagementRealm** security realm. It uses a file called **mgmt-users.properties** to store its configuration information.

Example 9.18. Default ManagementRealm

```
/host=master/core-service=management/security-realm=ManagementRealm/:read-
resource(recursive=true,proxies=false,include-runtime=false,include-
defaults=true)
{
    "outcome" => "success",
    "result" => {
        "authorization" => undefined,
        "server-identity" => undefined,
        "authentication" => {"properties" => {
            "path" => "mgmt-users.properties",
            "plain-text" => false,
            "relative-to" => "jboss.domain.config.dir"
        }}
    }
}
```

Write a Security Realm

The following commands create a new security realm called **TestRealm** and set the name and directory for the relevant properties file.

Example 9.19. Writing a Security Realm

```
/host=master/core-service=management/security-realm=TestRealm/:add
/host=master/core-service=management/security-
realm=TestRealm/authentication=properties/:write-
attribute(name=path,value=TestUsers.properties)
/host=master/core-service=management/security-
realm=TestRealm/authentication=properties/:write-attribute(name=relative-
to,value=jboss.domain.config.dir)
```

Apply a Security Realm to the Management Interface

After adding a security realm, supply its name as a reference to the Management Interface.

Example 9.20. Add a Security Realm to a Management Interface

```
host=master/core-service=management/management-interface=http-interface/:write-
attribute(name=security-realm,value=TestRealm)
```

Changes to management interfaces take effect after JBoss Enterprise Application Platform is restarted.

[Report a bug](#)

9.8. Network Security

9.8.1. Secure the Management Interfaces

Summary

In a test environment, it is typical to run JBoss Enterprise Application Platform 6 with no security layer on the management interfaces, comprised of the Management Console, Management CLI, and any other API implementation. This allows for rapid development and configuration changes.

In addition, the a silent authentication mode is present by default, allowing a local client on the host machine to connect to the Management CLI without requiring a username or password. This behaviour is a convenience for local users and Management CLI scripts, but it can be disabled if required. The procedure is described in the topic [Section 9.7.6, “Remove Silent Authentication from the Default Security Realm”](#).

When you begin testing and preparing your environment to move to production, it is vitally important to secure the management interfaces by at least the following methods:

- ▶ [Section 9.8.2, “Specify Which Network Interface the JBoss Enterprise Application Platform Uses”](#)
- ▶ [Section 9.8.3, “Configure Network Firewalls to Work with JBoss Enterprise Application Platform 6”](#)

[Report a bug](#)

9.8.2. Specify Which Network Interface the JBoss Enterprise Application Platform Uses

Overview

Isolating services so that they are accessible only to the clients who need them increases the security of your network. The JBoss Enterprise Application Platform includes two interfaces in its default configuration, both of which bind to the IP address **127.0.0.1**, or **localhost**, by default. One of the interfaces is called **management**, and is used by the Management Console, CLI, and API. The other is called **public**, and is used to deploy applications. These interfaces are not special or significant, but are provided as a starting point.

The **management** interface uses ports 9990 and 9999 by default, and the **public** interface uses port 8080, or port 8443 if you use HTTPS.

You can change the IP address of the management interface, public interface, or both.



Be cautious when exposing the management interfaces.

If you expose the management interfaces to other network interfaces which are accessible from remote hosts, be aware of the security implications. Most of the time, it is not advisable to provide remote access to the management interfaces.

1. **Stop the JBoss Enterprise Application Platform.**

Stop the JBoss Enterprise Application Platform by sending an interrupt in the appropriate way for your operating system. If you are running the JBoss Enterprise Application Platform as a foreground application, the typical way to do this is to press **Ctrl+C**.

2. **Restart the JBoss Enterprise Application Platform, specifying the bind address.**

Use the **-b** command-line switch to start the JBoss Enterprise Application Platform on a specific interface.

Example 9.21. Specify the public interface.

```
EAP_HOME/bin/domain.sh -b 10.1.1.1
```

Example 9.22. Specify the management interface.

```
EAP_HOME/bin/domain.sh -bmanagement=10.1.1.1
```

Example 9.23. Specify different addresses for each interface.

```
EAP_HOME/bin/domain.sh -bmanagement=127.0.0.1 -b 10.1.1.1
```

Example 9.24. Bind the public interface to all network interfaces.

```
EAP_HOME/bin/domain.sh -b 0.0.0.0
```

It is possible to edit your XML configuration file directly, to change the default bind addresses. However, if you do this, you will no longer be able to use the `-b` command-line switch to specify an IP address at run-time, so this is not recommended. If you do decide to do this, be sure to stop the JBoss Enterprise Application Platform completely before editing the XML file.

[Report a bug](#)

9.8.3. Configure Network Firewalls to Work with JBoss Enterprise Application Platform 6

Overview

Most production environments use firewalls as part of an overall network security strategy. If you need multiple server instances to communicate with each other or with external services such as web servers or databases, your firewall needs to take this into account. A well-managed firewall only opens the ports which are necessary to operation, and limits access to the ports to specific IP addresses, subnets, and network protocols.

A full discussion of firewalls is out of the scope of this documentation.

Prerequisites

- ▶ Determine the ports you need to open. Refer to [Section 9.8.4, "Network Ports Used By JBoss Enterprise Application Platform 6"](#) to determine the list of ports for your situation.
- ▶ An understanding of your firewall software is required. This procedure uses the `system-config-firewall` command in Red Hat Enterprise Linux 6. Microsoft Windows Server includes a built-in firewall, and several third-party firewall solutions are available for each platform.

Assumptions

This procedure configures a firewall in an environment with the following assumptions:

- ▶ The operating system is Red Hat Enterprise Linux 6.
- ▶ JBoss Enterprise Application Platform 6 runs on host **10.1.1.2**. Optionally, the server has its own firewall.
- ▶ The network firewall server runs on host **10.1.1.1** on interface **eth0**, and has an external interface **eth1**.
- ▶ You want traffic on port 5445 (a port used by JMS) forwarded to JBoss Enterprise Application Platform 6. No other traffic should be allowed through the network firewall.

Procedure 9.8. Task

1. Log into the Management Console.

Log into the Management Console. By default, it runs on <http://localhost:9990/console/>.

2. Managed Domain: Determine the socket binding group your server group uses.

Each server group uses a socket binding group, which is a collection of socket bindings. A socket

binding is a name-value pair of port name and number.

To determine which socket binding group your server groups, click the **Server Groups** label at the top right side of the screen. Then click the name of your server group in the **Available server group configurations** table. The **Server attributes** area at the bottom of the screen is populated with the profile and socket binding group used by the server group.

3. Determine the socket bindings used by the socket binding group.

Click the **Profiles** label at the top right of the Management Console. At the left-hand side of the screen, a series of menus is shown. The bottom menu heading is **General Configuration**. Click the **Socket Binding Groups** item below this heading. The **Socket Binding Declarations** screen appears. Initially, the **standard-sockets** group is shown. You can choose a different group by selecting it from the combo box on the right-hand side.

 **Note**

If you use a standalone server, it has only one socket binding group.

The list of socket names and ports is shown, six values per page. You can go through the pages by using the arrow navigation below the table.

4. Determine the ports you need to open.

Depending on the function of the particular port and the needs of your environment, some of the ports may need to be accessible across your firewall. If you are unsure of the purpose of a socket binding, refer to [Section 9.8.4. "Network Ports Used By JBoss Enterprise Application Platform 6"](#) for a list of the default socket bindings and their purposes.

5. Configure your firewall to forward traffic to JBoss Enterprise Application Platform 6.

Perform these steps to configure your network firewall to allow traffic on the desired port.

- a. Log into your firewall machine and access a command prompt, as the root user.
- b. Issue the command **system-config-firewall** to launch the firewall configuration utility. A GUI or command-line utility launches, depending on the way you are logged into the firewall system. This task makes the assumption that you are logged in via SSH and using the command-line interface.
- c. Use the **TAB** key on your keyboard to navigate to the **Customize** button, and press the **ENTER** key. The **Trusted Services** screen appears.
- d. Do not change any values, but use the **TAB** key to navigate to the **Forward** button, and press **ENTER** to advance to the next screen. The **Other Ports** screen appears.
- e. Use the **TAB** key to navigate to the **<Add>** button, and press **ENTER**. The **Port and Protocol** screen appears.
- f. Enter **5445** in the **Port / Port Range** field, then use the **TAB** key to move to the **Protocol** field, and enter **tcp**. Use the **TAB** key to navigate to the **OK** button, and press **ENTER**.
- g. Use the **TAB** key to navigate to the **Forward** button until you reach the **Port Forwarding** screen.
- h. Use the **TAB** key to navigate to the **<Add>** button, and press the **ENTER** key.
- i. Fill in the following values to set up port forwarding for port 5445.
 - ▶ Source interface: eth1
 - ▶ Protocol: tcp
 - ▶ Port / Port Range: 5445
 - ▶ Destination IP address: 10.1.1.2
 - ▶ Port / Port Range: 5445

Use the **TAB** key to navigate to the **OK** button, and press **ENTER**.

- j. Use the **TAB** key to navigate to the **Close** button, and press **ENTER**.
- k. Use the **TAB** key to navigate to the **OK** button, and press **ENTER**. To apply the changes, read the warning and click **Yes**.

6. Configure a firewall on your JBoss Enterprise Application Platform 6 host.

Some organizations choose to configure a firewall on the JBoss Enterprise Application Platform 6 server itself, and close all ports that are not necessary for its operation. Consult [Section 9.8.4. "Network Ports Used By JBoss Enterprise Application Platform 6"](#) and determine which ports to open, then close the rest. The default configuration of Red Hat Enterprise Linux 6 closes all ports except 22 (used for Secure Shell (SSH) and 5353 (used for multicast DNS). While you are configuring ports, make sure you have physical access to your server so that you do not

inadvertently lock yourself out.

Result

Your firewall is configured to forward traffic to your internal JBoss Enterprise Application Platform 6 server in the way you specified in your firewall configuration. If you chose to enable a firewall on your server, all ports are closed except the ones needed to run your applications.

[Report a bug](#)

9.8.4. Network Ports Used By JBoss Enterprise Application Platform 6

The ports used by the JBoss Enterprise Application Platform 6 default configuration depend on several factors:

- ▶ Whether you use a Managed Domain or Standalone Server configuration.
- ▶ Whether your server groups use one of the default socket binding groups, or a custom group.
- ▶ The requirements of your individual deployments.



Numerical port offsets

A numerical port offset can be configured, to alleviate port conflicts when you run multiple servers on the same physical server. If your server uses a numerical port offset, add the offset to the default port number for its server group's socket binding group. For instance, if the HTTP port of the socket binding group is 8080, and your server uses a port offset of 100, its HTTP port is 8180.

Unless otherwise stated, the ports use the TCP protocol.

The default socket binding groups

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

Table 9.2. Reference of the default socket bindings

Name	Port	Multicast Port	Description	full-ha-sockets	full-sockets	ha-socket	standar-d-socket
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.	Yes	Yes	Yes	Yes
http	8080		The default port for deployed web applications.	Yes	Yes	Yes	Yes
https	8443		SSL-encrypted connection between deployed web applications and clients.	Yes	Yes	Yes	Yes
jacorb	3528		CORBA services for JTS transactions and other ORB-dependent services.	Yes	Yes	No	No
jacorb-ssl	3529		SSL-encrypted CORBA services.	Yes	Yes	No	No
jgroup-s-diagonistics	7500		Multicast. Used for peer discovery in HA clusters.	Yes	No	Yes	No
jgroup-s-mping	45700		Multicast. Used to discover initial membership in a HA cluster.	Yes	No	Yes	No
jgroup-s-tcp	7600		Unicast peer discovery in HA clusters using TCP.	Yes	No	Yes	No
jgroup-s-tcp-fd	57600		Used for HA failure detection over TCP.	Yes	No	Yes	No
jgroup-s-udp	55200	45688	Unicast peer discovery in HA clusters using UDP.	Yes	No	Yes	No
jgroup-s-udp-fd	54200		Used for HA failure detection over UDP.	Yes	No	Yes	No
messaging	5445		JMS service.	Yes	Yes	No	No
messaging-group			Referenced by HornetQ JMS broadcast and discovery groups.	Yes	Yes	No	No
messaging-throughput	5455		Used by JMS Remoting.	Yes	Yes	No	No
mod_cluster	23364		Multicast port for communication between the JBoss Enterprise Application Platform and the HTTP load balancer.	Yes	No	Yes	No
osgi-http	8090		Used by internal components which use the OSGi	Yes	Yes	Yes	Yes

		subsystem.				
remote ng	4447	Used for remote EJB invocation.	Yes	Yes	Yes	Yes
txn-recovery-environment	4712	The JTA transaction recovery manager.	Yes	Yes	Yes	Yes
txn-status-manager	4713	The JTA / JTS transaction manager.	Yes	Yes	Yes	Yes

Management Ports

In addition to the socket binding groups, each host controller opens two more ports for management purposes:

- ▶ 9990 - The Web Management Console port
- ▶ 9999 - The port used by the Management Console and Management API

[Report a bug](#)

9.9. Java Security Manager

9.9.1. About the Java Security Manager

Java Security Manager

The Java Security Manager is a class that manages the external boundary of the Java Virtual Machine (JVM) sandbox, controlling how code executing within the JVM can interact with resources outside the JVM. When the Java Security Manager is activated, the Java API checks with the security manager for approval before executing a wide range of potentially unsafe operations.

The Security Manager uses a security policy to determine whether a given action will be permitted or denied.

Security Policy

A set of defined permissions for different classes of code. The Java Security Manager compares actions requested by applications against the security policy. If an action is allowed by the policy, the Security Manager will permit that action to take place. If the action is not allowed by the policy, the Security Manager will deny that action. The security policy can define permissions based on the location of code or on the code's signature.

The Security Manager and the security policy used are configured using the Java Virtual Machine options `java.security.manager` and `java.security.policy`.

[Report a bug](#)

9.9.2. Run JBoss Enterprise Application Platform Within the Java Security Manager

To specify a Java Security Manager policy, you need to edit the Java options passed to the domain or server instance during the bootstrap process. For this reason, you cannot pass the parameters as options to the `domain.sh` or `standalone.sh` scripts. The following procedure guides you through the steps of configuring your instance to run within a Java Security Manager policy.

Prerequisites

- ▶ Before you follow this procedure, you need to write a security policy, using the `policytool` command which is included with your Java Development Kit (JDK). This procedure assumes that your policy is located at `EAP_HOME/bin/server.policy`.
- ▶ The domain or standalone server must be completely stopped before you edit any configuration files.

Perform the following procedure for each physical host or instance in your domain, if you have domain members spread across multiple systems.

Procedure 9.9. Task

1. Edit the configuration file.

Open the configuration file for editing. This file is located in one of two places, depending on whether you use a managed domain or standalone server. This is not the executable file used to start the server or domain.

A. Managed Domain

`EAP_HOME/bin/domain.conf`

B. Standalone Server

`EAP_HOME/bin/standalone.conf`

2. Add the Java options at the end of the file.

Add the following line to a new line at the very end of the file. You can modify the -

Djava.security.policy value to specify the exact location of your security policy. It should go onto one line only, with no line break. You can modify the **-Djava.security.debug** to log more or less information, by specifying the debug level. The most verbose is **failure,access,policy**.

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -Djboss.home.dir=$PWD/... -  
Djava.security.policy==$PWD/server.policy -Djava.security.debug=failure"
```

3. Start the domain or server.

Start the domain or server as normal.

[Report a bug](#)

9.9.3. About Java Security Manager Policies

The Java Security Manager uses a security policy to determine whether a given action will be permitted or denied.

The security policy is a set of defined permissions for different classes of code. The Java Security Manager compares actions requested by applications against the security policy. If an action is allowed by the policy, the Java Security Manager will permit that action to take place. If the action is not allowed by the policy, the Java Security Manager will deny that action. The security policy can define permissions based on the location of code or on the code's signature.

The Java Security Manager and the security policy used are configured using the Java Virtual Machine options **java.security.manager** and **java.security.policy**.

[Report a bug](#)

9.9.4. Write a Java Security Manager Policy

Introduction

An application called **policytool** is included with most JDK and JRE distributions, for the purpose of creating and editing Java Security Manager security policies. Detailed information about **policytool** is linked from <http://docs.oracle.com/javase/6/docs/technotes/tools/>.

Basic Information

A security policy consists of the following configuration elements:

CodeBase

The URL location (excluding the host and domain information) where the code originates from. This parameter is optional.

SignedBy

The alias used in the keystore to reference the signer whose private key was used to sign the code. This can be a single value or a comma-separated list of values. This parameter is optional. If omitted, presence or lack of a signature has no impact on the Java Security Manager.

Principals

A list of principal_type/principal_name pairs, which must be present within the executing thread's principal set. The Principals entry is optional. If it is omitted, it signifies "any principals".

Permissions

A permission is the access which is granted to the code. Many permissions are provided as part of the Java Enterprise Edition 6 (Java EE 6) specification. This document only covers additional permissions which are provided by JBoss Enterprise Application Platform.

Procedure 9.10. Task

1. Start policytool.

Start the **policytool** tool in one of the following ways.

A. Red Hat Enterprise Linux

From your GUI or a command prompt, run **/usr/bin/policytool**.

B. Microsoft Windows Server

Run **policytool.exe** from your Start menu or from the **bin** of your Java installation. The location can vary.

2. Create a new policy.

To create a new policy, select **Add Policy Entry**. Add the parameters you need, then click **Done**.

3. Edit an existing policy

Select the policy from the list of existing policies, and select the **Edit Policy Entry** button. Edit the parameters as needed.

4. Delete an existing policy.

Select the policy from the list of existing policies, and select the **Delete Policy Entry** button.

Permission Specific to JBoss Enterprise Application Platform

org.jboss.security.SecurityAssociation.getPrincipalInfo

Provides access to the **org.jboss.security.SecurityAssociationgetPrincipal()** and **getCredential()** methods. The risk involved with using this runtime permission is the ability to see the current thread caller and credentials.

org.jboss.security.SecurityAssociation.getSubject

Provides access to the **org.jboss.security.SecurityAssociationgetSubject()** method.

org.jboss.security.SecurityAssociation.setPrincipalInfo

Provides access to the **org.jboss.security.SecurityAssociationsetPrincipal()**, **setCredential()**, **setSubject()**, **pushSubjectContext()**, and **popSubjectContext()** methods. The risk involved with using this runtime permission is the ability to set the current thread caller and credentials.

org.jboss.security.SecurityAssociation.setServer

Provides access to the **org.jboss.security.SecurityAssociationsetServer** method. The risk involved with using this runtime permission is the ability to enable or disable multi-thread storage of the caller principal and credential.

org.jboss.security.SecurityAssociation.setRunAsRole

Provides access to the **org.jboss.security.SecurityAssociationpushRunAsRole**, **popRunAsRole**, **pushRunAsIdentity**, and **popRunAsIdentity** methods. The risk involved with using this runtime permission is the ability to change the current caller run-as role principal.

org.jboss.security.SecurityAssociation.accessContextInfo

Provides access to the **org.jboss.security.SecurityAssociationaccessContextInfo**, and **accessContextInfo** getter and setter methods. This allows you to both set and get the

current security context info.

org.jboss.naming.JndiPermission

Provides special permissions to files and directories in a specified JNDI tree path, or recursively to all files and subdirectories. A JndiPermission consists of a pathname and a set of valid permissions related to the file or directory.

The available permissions include:

- » bind
- » rebind
- » unbind
- » lookup
- » list
- » listBindings
- » createSubcontext
- » all

Pathnames ending in /* indicate that the specified permissions apply to all files and directories of the pathname. Pathnames ending in /- indicate recursive permissions to all files and subdirectories of the pathname. Pathnames consisting of the special token <>ALL BINDINGS>> matches any file in any directory.

org.jboss.security.srp.SRPPermission

A custom permission class for protecting access to sensitive SRP information like the private session key and private key. This permission does not have any actions defined. The **getSessionKey** target provides access to the private session key which results from the SRP negotiation. Access to this key allows you to encrypt and decrypt messages that have been encrypted with the session key.

org.hibernate.secure.HibernatePermission

This permission class provides basic permissions to secure Hibernate sessions. The target for this property is the entity name. The available actions include:

- » insert
- » delete
- » update
- » read
- » * (all)

org.jboss.metadata.spi.stack.MetaDataStackPermission

Provides a custom permission class for controlling how callers interact with the metadata stack. The available permissions are:

- » modify
- » push (onto the stack)
- » pop (off the stack)
- » peek (onto the stack)
- » * (all)

org.jboss.config.spi.ConfigurationPermission

Secures setting of configuration properties. Defines only permission target names, and no actions. The targets for this property include:

- » <property name> (the property this code has permission to set)
- » * (all properties)

org.jboss.kernel.KernelPermission

Secures access to the kernel configuration. Defines only permission target names and no actions. The targets for this property include:

- ▶ access (to the kernel configuration)
- ▶ configure (implies access)
- ▶ * (all)

org.jboss.kernel.plugins.util.KernelLocatorPermission

Secures access to the kernel. Defines only permission target names and no actions. The targets for this property include:

- ▶ kernel
- ▶ * (all)

[Report a bug](#)

9.9.5. Debug Security Manager Policies

You can enable debugging information to help you troubleshoot security policy-related issues. The **java.security.debug** option configures the level of security-related information reported. The command **java -Djava.security.debug=help** will produce help output with the full range of debugging options. Setting the debug level to **all** is useful when troubleshooting a security-related failure whose cause is completely unknown, but for general use it will produce too much information. A sensible general default is **access:failure**.

Procedure 9.11. Enable general debugging

- ▶ **This procedure will enable a sensible general level of security-related debug information.**

Add the following line to the server configuration file.

- If the JBoss Enterprise Application Platform instance is running in a managed domain, the line is added to the **bin/domain.conf** file for Linux or the **bin\domain.conf.bat** file for Windows.
- If the JBoss Enterprise Application Platform instance is running as a standalone server, the line is added to the **bin/standalone.conf** file for Linux, or the **bin\standalone.conf.bat** file for Windows.

Linux

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.debug=access:failure"
```

Windows

```
JAVA_OPTS="%JAVA_OPTS% -Djava.security.debug=access:failure"
```

Result

A general level of security-related debug information has been enabled.

[Report a bug](#)

9.10. Application Security

9.10.1. Enabling/Disabling Descriptor Based Property Replacement

Summary

Finite control over descriptor property replacement was introduced in **jboss-as-ee_1_1.xsd**. This task covers the steps required to configure descriptor based property replacement.

Prerequisites

- ▶ [Section 2.6.1, “Start JBoss Enterprise Application Platform 6”](#).
- ▶ [Section 3.3.2, “Launch the Management CLI”](#).

Descriptor based property replacement flags have boolean values:

- ▶ When set to **true**, property replacements are enabled.

- When set to **false**, property replacements are disabled.

Procedure 9.12. jboss-descriptor-property-replacement

jboss-descriptor-property-replacement is used to enable or disable property replacement in the following descriptors:

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

The default value for **jboss-descriptor-property-replacement** is **true**.

- In the Management CLI, run the following command to determine the value of **jboss-descriptor-property-replacement**:

```
/subsystem=ee:read-attribute(name="jboss-descriptor-property-replacement")
```

- Run the following command to configure the behavior:

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

Procedure 9.13. spec-descriptor-property-replacement

spec-descriptor-property-replacement is used to enable or disable property replacement in the following descriptors:

- ejb-jar.xml**
- persistence.xml**

The default value for **spec-descriptor-property-replacement** is **false**.

- In the Management CLI, run the following command to confirm the value of **spec-descriptor-property-replacement**:

```
/subsystem=ee:read-attribute(name="spec-descriptor-property-replacement")
```

- Run the following command to configure the behavior:

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

Result

The descriptor based property replacement tags have been successfully configured.

[Report a bug](#)

9.11. Password Vaults for Sensitive Strings

9.11.1. About Securing Sensitive Strings in Clear-Text Files

Web applications and other deployments often include clear-text files, such as XML deployment descriptors, which include sensitive information such as passwords and other sensitive strings. JBoss Enterprise Application Platform includes a password vault mechanism which enables you to encrypt sensitive strings and store them in an encrypted keystore. The vault mechanism manages decrypting the strings for use with security domains, security realms, or other verification systems. This provides an extra layer of security. The mechanism relies upon tools that are included in all supported Java Development Kit (JDK) implementations.

[Report a bug](#)

9.11.2. Create a Java Keystore to Store Sensitive Strings

Prerequisites

- ▶ The **keytool** command must be available to use. It is provided by the Java Runtime Environment (JRE). Locate the path for the file. In Red Hat Enterprise Linux, it is installed to **/usr/bin/keytool**.

Procedure 9.14. Task

1. Create a directory to store your keystore and other encrypted information.

Create a directory to hold your keystore and other important information. The rest of this procedure assumes that the directory is **/home/*USER/vault/***.

2. Determine the parameters to use with keytool.

Determine the following parameters:

alias

The alias is a unique identifier for the vault or other data stored in the keystore. The alias in the example command at the end of this procedure is **vault**. Aliases are case-insensitive.

keyalg

The algorithm to use for encryption. The default is **DSA**. The example in this procedure uses **RSA**. Check the documentation for your JRE and operating system to see which other choices may be available to you.

keysize

The size of an encryption key impacts how difficult it is to decrypt through brute force. The default size of keys is 1024. It must be between 512 and 1024, and a multiple of 64. The example in this procedure uses **1024**.

keystore

The keystore a database which holds encrypted information and the information about how to decrypt it. If you do not specify a keystore, the default keystore to use is a file called **.keystore** in your home directory. The first time you add data to a keystore, it is created. The example in this procedure uses the **vault.keystore** keystore.

The **keystore** command has many other options. Refer to the documentation for your JRE or your operating system for more details.

3. Determine the answers to questions the keystore command will ask.

The **keystore** needs the following information in order to populate the keystore entry:

Keystore password

When you create a keystore, you must set a password. In order to work with the keystore in the future, you need to provide the password. Create a strong password that you will remember. The keystore is only as secure as its password and the security of the file system and operating system where it resides.

Key password (optional)

In addition to the keystore password, you can specify a password for each key it holds. In order to use such a key, the password needs to be given each time it is used. Usually, this facility is not used.

First name (given name) and last name (surname)

This, and the rest of the information in the list, helps to uniquely identify the key and place it into a hierarchy of other keys. It does not necessarily need to be a name at all, but it should be two words, and must be unique to the key. The example in this procedure uses **Accounting Administrator**. In directory terms, this becomes the *common name* of the certificate.

Organizational unit

This is a single word that identifies who uses the certificate. It may be the application or the business unit. The example in this procedure uses **AccountingServices**.

Typically, all keystores used by a group or application use the same organizational unit.

Organization

This is usually a single-word representation of your organization's name. This typically remains the same across all certificates used by an organization. This example uses **MyOrganization**.

City or municipality

Your city.

State or province

Your state or province, or the equivalent for your locality.

Country

The two-letter code for your country.

All of this information together will create a hierarchy for your keystores and certificates, ensuring that they use a consistent naming structure but are unique.

4. Run the keytool command, supplying the information that you gathered.

Example 9.25. Example input and output of keystore command

```
$ keytool -genkey -alias vault -keyalg RSA -keysize 1024 -keystore
/home/USER/vault/vault.keystore
Enter keystore password: vault22
Re-enter new password:vault22
What is your first and last name?
[Unknown]: Accounting Administrator
What is the name of your organizational unit?
[Unknown]: AccountingServices
What is the name of your organization?
[Unknown]: MyOrganization
What is the name of your City or Locality?
[Unknown]: Raleigh
What is the name of your State or Province?
[Unknown]: NC
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=Accounting Administrator, OU=AccountingServices, O=MyOrganization,
L=Raleigh, ST=NC, C=US correct?
[no]: yes

Enter key password for <vault>
(RTURN if same as keystore password):
```

Result

A file named **vault.keystore** is created in the **/home/USER/vault/** directory. It stores a single key, called **vault**, which will be used to store encrypted strings, such as passwords, for the JBoss Enterprise Application Platform.

[Report a bug](#)

9.11.3. Mask the Keystore Password and Initialize the Password Vault

Prerequisites

- » [Section 9.11.2, "Create a Java Keystore to Store Sensitive Strings"](#)
- » The **EAP_HOME/bin/vault.sh** application needs to be accessible via a command-line interface.

1. Run the vault.sh command.

Run **EAP_HOME/bin/vault.sh**. Start a new interactive session by typing **0**.

2. Enter the directory where encrypted files will be stored.

This directory should be reasonably secure, but the JBoss Enterprise Application Platform needs to be able to access it. If you followed [Section 9.11.2, "Create a Java Keystore to Store Sensitive Strings"](#), your keystore is in a directory called **vault/** in your home directory. This example uses the directory **/home/USER/vault/**.

 **Include the trailing slash on the directory name.**

Do not forget to include the trailing slash on the directory name. Either use **/** or ****, depending on your operating system.

3. Enter the path to the keystore.

Enter the full path to the keystore file. This example uses `/home/USER/vault/vault.keystore`.

4. Encrypt the keystore password.

The following steps encrypt the keystore password, so that you can use it in configuration files and applications securely.

a. Enter the keystore password.

When prompted, enter the keystore password.

b. Enter a salt value.

Enter an 8-character salt value. The salt value, together with the iteration count (below), are used to create the hash value.

c. Enter the iteration count.

Enter a number for the iteration count.

d. Make a note of the masked password information.

The masked password, the salt, and the iteration count are printed to standard output.

Make a note of them in a secure location. An attacker could use them to decrypt the password.

e. Enter the alias of the vault.

When prompted, enter the alias of the vault. If you followed [Section 9.11.2, “Create a Java Keystore to Store Sensitive Strings”](#) to create your vault, the alias is `vault`.

5. Exit the interactive console.

Type `exit` to exit the interactive console.

Result

Your keystore password has been masked for use in configuration files and deployments. In addition, your vault is fully configured and ready to use.

[Report a bug](#)

9.11.4. Configure the JBoss Enterprise Application Platform to Use the Password Vault

Overview

Before you can mask passwords and other sensitive attributes in configuration files, you need to make the JBoss Enterprise Application Platform aware of the password vault which stores and decrypts them. Follow this procedure to enable this functionality.

Prerequisites

- ▶ [Section 9.11.2, “Create a Java Keystore to Store Sensitive Strings”](#)
- ▶ [Section 9.11.3, “Mask the Keystore Password and Initialize the Password Vault”](#)

Procedure 9.15. Task

1. Determine the correct values for the command.

Determine the values for the following parameters, which are determined by the commands used to create the keystore itself. For information on creating a keystore, refer to the following topics: [Section 9.11.2, “Create a Java Keystore to Store Sensitive Strings”](#) and [Section 9.11.3, “Mask the Keystore Password and Initialize the Password Vault”](#).

Parameter	Description
KEYSTORE_URL	The file system path or URI of the keystore file, usually called something like <code>vault.keystore</code>
KEYSTORE_PASSWORD	The password used to access the keystore. This value should be masked.
KEYSTORE_ALIAS	The name of the keystore.
SALT	The salt used to encrypt and decrypt keystore values.
ITERATION_COUNT	The number of times the encryption algorithm is run.
ENC_FILE_DIR	The path to the directory from which the

host (managed domain only)	keystore commands are run. Typically the directory containing the password vault.
	The name of the host you are configuring

2. Use the Management CLI to enable the password vault.

Run one of the following commands, depending on whether you use a managed domain or standalone server configuration. Substitute the values in the command with the ones from the first step of this procedure.

A. Managed Domain

```
/host=YOUR_HOST/core-service=vault:add(vault-options=[("KEYSTORE_URL" => "PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"), ("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"), ("ITERATION_COUNT" => "ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

B. Standalone Server

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" => "PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"), ("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"), ("ITERATION_COUNT" => "ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

The following is an example of the command with hypothetical values:

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" => "/home/user/vault/vault.keystore"), ("KEYSTORE_PASSWORD" => "MASK-3y28rcZlcKR"), ("KEYSTORE_ALIAS" => "vault"), ("SALT" => "12438567"), ("ITERATION_COUNT" => "50"), ("ENC_FILE_DIR" => "/home/user/vault/"))]
```

Result

The JBoss Enterprise Application Platform is configured to decrypt masked strings using the password vault. To add strings to the vault and use them in your configuration, refer to the following topic:

[Section 9.11.5, “Store and Retrieve Encrypted Sensitive Strings in the Java Keystore”](#).

[Report a bug](#)

9.11.5. Store and Retrieve Encrypted Sensitive Strings in the Java Keystore

Overview

Including passwords and other sensitive strings in plain-text configuration files is insecure. The JBoss Enterprise Application Platform includes the ability to store and mask these sensitive strings in an encrypted keystore, and use masked values in configuration files.

Prerequisites

- ▶ [Section 9.11.2, “Create a Java Keystore to Store Sensitive Strings”](#)
- ▶ [Section 9.11.3, “Mask the Keystore Password and Initialize the Password Vault”](#)
- ▶ [Section 9.11.4, “Configure the JBoss Enterprise Application Platform to Use the Password Vault”](#)
- ▶ The `EAP_HOME/bin/vault.sh` application needs to be accessible via a command-line interface.

Procedure 9.16. Task

1. Run the `vault.sh` command.

Run `EAP_HOME/bin/vault.sh`. Start a new interactive session by typing `0`.

2. Enter the directory where encrypted files will be stored.

If you followed [Section 9.11.2, “Create a Java Keystore to Store Sensitive Strings”](#), your keystore is in a directory called `vault/` in your home directory. In most cases, it makes sense to store all of your encrypted information in the same place as the key store. This example uses the directory `/home/USER/vault/`.

3. Enter the path to the keystore.

Enter the full path to the keystore file. This example uses **/home/*USER/vault/vault.keystore***.

4. Enter the keystore password, vault name, salt, and iteration count.

When prompted, enter the keystore password, vault name, salt, and iteration count. A handshake is performed.

5. Select the option to store a password.

Select option **0** to store a password or other sensitive string.

6. Enter the value.

When prompted, enter the value twice. If the values do not match, you are prompted to try again.

7. Enter the vault block.

Enter the vault block, which is a container for attributes which pertain to the same resource. An example of an attribute name would be **ds_ExampleDS**. This will form part of the reference to the encrypted string, in your datasource or other service definition.

8. Enter the attribute name.

Enter the name of the attribute you are storing. An example attribute name would be **password**.

Result

A message such as the one below shows that the attribute has been saved.

Attribute Value for (ds_ExampleDS, password) saved

9. Make note of the information about the encrypted string.

A message prints to standard output, showing the vault block, attribute name, shared key, and advice about using the string in your configuration. Make note of this information in a secure location. Example output is shown below.

```
*****
Vault Block:ds_ExampleDS
Attribute Name:password
Shared
Key:N2NhZDYz0T MtNWE00S00ZGQ0LWE4MmEtMWN1MDMyNDdmNmI2TE10RV9CUkVBS3ZhdWx0
Configuration should be done as follows:
VAULT::ds_ExampleDS::password::N2NhZDYz0T MtNWE00S00ZGQ0LWE4MmEtMWN1MDMyNDdmN
mI2TE10RV9CUkVBS3ZhdWx0
*****
```

10. Use the encrypted string in your configuration.

Use the string from the previous step in your configuration, in place of a plain-text string. A datasource using the encrypted password above is shown below.

```
...
<subsystem xmlns="urn:jboss:domain:datasources:1.0">
    <datasources>
        <datasource jndi-name="java:jboss/datasources/ExampleDS" enabled="true"
use-java-context="true" pool-name="H2DS">
            <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
            <driver>h2</driver>
            <pool></pool>
            <security>
                <user-name>sa</user-name>

            <password>VAULT::ds_ExampleDS::password::N2NhZDYz0T MtNWE00S00ZGQ0LWE4MmEtMWN1
MDMyNDdmNmI2TE10RV9CUkVBS3ZhdWx0</password>
                </security>
        </datasource>
        <drivers>
            <driver name="h2" module="com.h2database.h2">
                <xa-datasource-class>org.h2.jdbc.JdbcDataSource</xa-datasource-
class>
            </driver>
        </drivers>
    </datasources>
</subsystem>
...
```

You can use an encrypted string anywhere in your domain or standalone configuration file. After you store your string in the keystore, use the following syntax to replace any clear-text string with an encrypted one.

```
${VAULT::<replaceable>VAULT_BLOCK</replaceable>::<replaceable>ATTRIBUTE_NAME</replaceable>::<replaceable>ENCRYPTED_VALUE</replaceable>}
```

Here is a sample real-world value, where the vault block is **ds_ExampleDS** and the attribute is **password**.

```
<password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE00S00ZGQ0LWE4MmEtMW  
N1MDMyNDdmNmI2TE10RV9CUkVBS3ZhdWx0}</password>
```

[Report a bug](#)

9.11.6. Store and Resolve Sensitive Strings In Your Applications

Overview

Configuration elements of the JBoss Enterprise Application Platform support the ability to resolve encrypted strings against values stored in a Java Keystore, via the Security Vault mechanism. You can add support for this feature to your own applications.

First, add the password to the vault. Second, replace the clear-text password with the one stored in the vault. You can use this method to obscure any sensitive string in your application.

Prerequisites

Before performing this procedure, make sure that the directory for storing your vault files exists. It does not matter where you place them, as long as the user who executes JBoss Enterprise Application Platform has permission to read and write the files. This example places the **vault/** directory into the **/home/*USER*/vault/** directory. The vault itself is a file called **vault.keystore** inside the **vault/** directory.

Example 9.26. Adding the Password String to the Vault

Add the string to the vault using the **EAP_HOME/bin/vault.sh** command. The full series of commands and responses is included in the following screen output. Values entered by the user are emphasized. Some output is removed for formatting. In Microsoft Windows, the name of the command is **vault.bat**. Note that in Microsoft Windows, file paths use the \ character as a directory separator, rather than the / character.

```
[user@host bin]$ ./vault.sh
*****
*** JBoss Vault ***
*****
Please enter a Digit:: 0: Start Interactive Session 1: Remove Interactive
Session 2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:/home/user/vault/
Enter Keystore URL:/home/user/vault/vault.keystore
Enter Keystore password: ...
Enter Keystore password again: ...
Values match
Enter 8 character salt:12345678
Enter iteration count as a number (Eg: 44):25

Enter Keystore Alias:vault
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit:: 0: Store a password 1: Check whether password exists
2: Exit
0
Task: Store a password
Please enter attribute value: sa
Please enter attribute value again: sa
Values match
Enter Vault Block:DS
Enter Attribute Name:thePass
Attribute Value for (DS, thePass) saved

Please make note of the following:
*****
Vault Block:DS
Attribute Name:thePass
Shared Key:OWY5M2I5NzctYzdkOS00MmZhLWExZGYtNjczM2U5ZGUy0WIxTE10RV9CUKVBS3ZhdWx0
Configuration should be done as follows:
VAULT::DS::thePass::OWY5M2I5NzctYzdkOS00MmZhLWExZGYtNjczM2U5ZGUy0WIxTE10RV9CUKVBS
3ZhdWx0
*****
```

Please enter a Digit:: 0: Store a password 1: Check whether password exists
2

The string that will be added to the Java code is the last value of the output, the line beginning with **VAULT**.

The following servlet uses the vaulted string instead of a clear-text password. The clear-text version is commented out so that you can see the difference.

Example 9.27. Servlet Using a Vaulted Password

```

package vaulterror.web;

import java.io.IOException;
import java.io.Writer;

import javax.annotation.Resource;
import javax.annotation.sql.DataSourceDefinition;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/*@DataSourceDefinition(
    name = "java:jboss/datasources/LoginDS",
    user = "sa",
    password = "sa",
    className = "org.h2.jdbcx.JdbcDataSource",
    url = "jdbc:h2:tcp://localhost/mem:test"
)*/
@DataSourceDefinition(
    name = "java:jboss/datasources/LoginDS",
    user = "sa",
    password =
"VAULT::DS::thePass::0WY5M2I5NzctYzdkOS00MmZhLWExZGYtNjczM2U5ZGUy0WIxTE10RV9CUkVB
S3ZhdWx0",
    className = "org.h2.jdbcx.JdbcDataSource",
    url = "jdbc:h2:tcp://localhost/mem:test"
)
@WebServlet(name = "MyTestServlet", urlPatterns = { "/my/" }, loadOnStartup = 1)
public class MyTestServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Resource(lookup = "java:jboss/datasources/LoginDS")
    private DataSource ds;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        Writer writer = resp.getWriter();
        writer.write((ds != null) + "");
    }
}

```

Your servlet is now able to resolve the vaulted string.

[Report a bug](#)

Chapter 10. Security Administration Reference

10.1. Included Authentication Modules

The following authentication modules are included in the JBoss Enterprise Application Platform. Some of these handle authorization as well as authentication. These usually include the word **Role** within the **Code** name.

When you configure these modules, use the **Code** value to refer to the module.

Authentication Modules

- ▶ [Table 10.1, "Client"](#)
- ▶ [Table 10.3, "Certificate"](#)
- ▶ [Table 10.5, "CertificateUsers"](#)
- ▶ [Table 10.7, "CertificateRoles"](#)
- ▶ [Table 10.9, "Database"](#)
- ▶ [Table 10.11, "DatabaseCertificate"](#)
- ▶ [Table 10.13, "Identity"](#)
- ▶ [Table 10.15, "Ldap"](#)
- ▶ [Table 10.17, "LdapExtended"](#)
- ▶ [Table 10.19, "RoleMapping"](#)
- ▶ [Table 10.21, "RunAs"](#)
- ▶ [Table 10.23, "Simple"](#)
- ▶ [Table 10.24, "ConfiguredIdentity"](#)
- ▶ [Table 10.26, "SecureIdentity"](#)
- ▶ [Table 10.28, "PropertiesUsers"](#)
- ▶ [Table 10.30, "SimpleUsers"](#)
- ▶ [Table 10.32, "LdapUsers"](#)
- ▶ [Table 10.33, "Kerberos"](#)
- ▶ [Table 10.35, "SPNEGOUsers"](#)
- ▶ [Table 10.37, "AdvancedLdap"](#)
- ▶ [Table 10.39, "AdvancedADLdap"](#)
- ▶ [Table 10.41, "UsersRoles"](#)
- ▶ [Custom Authentication Modules](#)

Table 10.1. Client

Code	client
Class	org.jboss.security.ClientLoginModule
Description	This login module is designed to establish caller identity and credentials when the JBoss Enterprise Application Platform is acting as a client. It should never be used as part of a security domain used for actual server authentication.

Table 10.2. Client Module Options

Option	Type	Default	Description
<code>multi-threaded</code>	true or false	false	Set to true if each thread has its own principal and credential storage. Set to false to indicate that all threads in the VM share the same identity and credential.
<code>password-stacking</code>	useFirstPass or false	false	Set to useFirstPass to indicate that this login module should look for information stored in the LoginContext to use as the identity. This option can be used when stacking other login modules with this one.
<code>>restore-login-identity</code>	true or false	false	Set to true if the identity and credential seen at the start of the login() method should be restored after the logout() method is invoked.

Table 10.3. Certificate

Code	certificate
Class	<code>org.jboss.security.auth.spi.BaseCertLoginModule</code>
Description	This login module is designed to authenticate users based on X509 Certificates . A use case for this is CLIENT -CERT authentication of a web application.

Table 10.4. Certificate Module Options

Option	Type	Default	Description
<code>securityDomain</code>	string	none	Name of the security domain that has the JSSE configuration for the truststore holding the trusted certificates.
<code>verifier</code>	Class	none	The class name of the <code>org.jboss.security.auth.certs.X509CertificateVerifier</code> to use for verification of the login certificate.

Table 10.5. CertificateUsers

Code	certificateUsers
Class	<code>org.jboss.security.auth.spi.UsersRoleLoginModule</code>
Description	Uses a properties resources. The first maps usernames to passwords, and the second maps usernames to roles.

Table 10.6. CertificateUsers Module Options

Option	Type	Default	Description
unauthenticatedId entity	A string	none	Defines the principal name that should be assigned to requests which contain no authentication information. This can allow unprotected servlets to invoke methods on EJBs that do not require a specific role. Such a principal has no associated roles and can only access either unsecured EJBs or EJB methods that are associated with the unchecked permission constraint.
password-stacking	useFirstPass or false	false	Set to useFirstPass to indicate that this login module should look for information stored in the LoginContext to use as the identity. This option can be used when stacking other login modules with this one.
hashAlgorithm	A string	none	The name of the java.security.MessageDigest algorithm to use to hash the password. There is no default so this option must be explicitly set to enable hashing. When hashAlgorithm is specified, the clear text password obtained from the CallbackHandler is hashed before it is passed to UsernamePasswordLoginModule.validatePassword as the inputPassword argument. The expectedPassword stored in the users.properties file must be comparably hashed. Refer to http://docs.oracle.com/javase/6/docs/api/java/security/MessageDigest.html for information on java.security.MessageDigest and the algorithms this class

hashEncoding	base64 or hex	base64	supports.
hashCharSet	A string	The default encoding set in the container's environment.	The encoding used to convert the clear-text password to a byte array.
usersProperties	The fully-qualified file path and name of a properties file or resource	users.properties	The file containing the mappings between users and passwords. Each property in the file has the format username=password .
rolesProperties	The fully-qualified file path and name of a properties file or resource	roles.properties	The file containing the mappings between users and roles. Each property in the file has the format username=role1,role2,...,roleN .
ignorePasswordCase	true or false	false	Whether the password comparison should ignore case. This is useful for hashed password encoding where the case of the hashed password is not significant.
principalClass	A fully-qualified classname.	none	A Principal implementation class which contains a constructor that takes a String argument for the principal name.
roleGroupSeparator	A single character	. (a single period)	The character used to separate the username from the role group name in the rolesGroup file.
defaultUsersProperties	string	defaultUsers.properties	Name of the resource or file to fall back to if the usersProperties file can't be found.
defaultRolesProperties	string	defaultRoles.properties	Name of the resource or file to fall back to if the rolesProperties file cannot be found.
hashUserPassword	true or false	true	Whether to hash the password entered by the user, when hashAlgorithm is specified. Defaults to true .
hashStorePassword	true or false	true	Whether the store password returned from getUsersPassword() should be hashed, when hashAlgorithm is

digestCallback	A fully-qualified classname.	none	specified. The class name of the org.jboss.crypto.digest.DigestCallback implementation that includes pre or post digest content such as salt values. Only used if hashAlgorithm is specified.
storeDigestCallback	A fully-qualified classname.	none	The class name of the org.jboss.crypto.digest.DigestCallback implementation that includes pre/post digest content like salts for hashing the store password. Only used if hashStorePassword is true and hashAlgorithm is specified.
callback.option.STRING	Various	none	All options prefixed with callback.option. are passed to the DigestCallback.init(Map) method. The input username is always passed in via the javax.security.auth.login.name option, and the input/store password is passed in via the javax.security.auth.login.password option to the digestCallback or storeDigestCallback .

Table 10.7. CertificateRoles

Code	CertificateRoles
Class	org.jboss.security.auth.spi.CertRolesLoginModule
Description	This login module extends the Certificate login module to add role mapping capabilities from a properties file. It takes all of the same options as the Certificate login module, and adds the following options.

Table 10.8. CertificateRoles Module Options

Option	Type	Default	Description
rolesProperties	A string	roles.properties	The name of the resource or file containing the roles to assign to each user. The role properties file must be in the format username=role1,role2 where the username is the DN of the certificate, escaping any = (equals) and space characters. The following example is in the correct format:
			<pre>CN\=unit-tests-client,\OU\=Red\ Hat\Inc.,\O\=Red\ Hat\ Inc.\ST\=North\Carolina,\C\=US=JBossAdmin</pre>
defaultRolesProperties	A string	defaultRoles.properties	Name of the resource or file to fall back to if the rolesProperties file cannot be found.
roleGroupSeparator	A single character	.	(a single period) Which character to use as the role group separator in the roleProperties file.

Table 10.9. Database

Code	Database
Class	org.jboss.security.auth.spi.DatabaseServerLoginModule
Description	<p>A JDBC-based login module that supports authentication and role mapping. It is based on two logical tables, with the following definitions.</p> <ul style="list-style-type: none"> ▶ Principals: <code>PrincipalID (text), Password (text)</code> ▶ Roles: <code>PrincipalID (text), Role (text), RoleGroup (text)</code>

Table 10.10. Database Module Options

Option	Type	Default	Description
dsJndiName	A JNDI resource	none	The name of the JNDI resource storing the authentication information. This option is required.
principalsQuery	A prepared SQL statement	select Password from Principals where PrincipalID=?	The prepared SQL query to obtain the information about the principal.
rolesQuery	A prepared SQL statement	select Role, RoleGroup from Roles where PrincipalID=?	The prepared SQL query to obtain the information about the roles.

Table 10.11. DatabaseCertificate

Code	DatabaseCertificate
Class	org.jboss.security.auth.spi.DatabaseC ertLoginModule
Description	This login module extends the Certificate login module to add role mapping capabilities from a database table. It has the same options plus these additional options:

Table 10.12. DatabaseCertificate Module Options

Option	Type	Default	Description
dsJndiName	A JNDI resource		The name of the JNDI resource storing the authentication information. This option is required.
rolesQuery	A prepared SQL statement	select Role,RoleGroup from Roles where PrincipalID=?	SQL prepared statement to be executed in order to map roles. It should be an equivalent to select Role, RoleGroup from Roles where PrincipalID=? , where Role is the role name and the RoleGroup column value should always be Roles . with capital R.
suspendResume	true or false	true	Whether any existing JTA transaction should be suspended during database operations.

Table 10.13. Identity

Code	Identity
Class	org.jboss.security.auth.spi.IdentityLoginModule
Description	Associates the principal specified in the module options with any subject authenticated against the module. The type of Principal class used is org.jboss.security.SimplePrincipal . If no principal option is specified a principal with the name of guest is used.

Table 10.14. Identity Module Options

Option	Type	Default	Description
principal	A string	guest	The name to use for the principal.
roles	A comma-separated list of strings	none	A comma-delimited list of roles which will be assigned to the subject.

Table 10.15. Ldap

Code	Ldap
Class	org.jboss.security.auth.spi.LdapLoginModule
Description	Authenticates against an LDAP server, when the username and password are stored in an LDAP server that is accessible using a JNDI LDAP provider. Many of the options are not required, because they are determined by the LDAP provider or the environment.

Table 10.16. Ldap Module Options

Option	Type	Default	Description
<code>java.naming.factory.initial</code>	class name	<code>com.sun.jndi.ldap.LdapCtxFactory</code>	<code>InitialContextFactory</code> implementation class name.
<code>java.naming.provider.url</code>	ldap:// URL	none	URL for the LDAP server.
<code>java.naming.security.authentication</code>	none, simple, or the name of a SASL mechanism	<code>simple</code>	The security level to use to bind to the LDAP server.
<code>java.naming.security.protocol</code>	A transport protocol	If unspecified, determined by the provider.	The transport protocol to use for secure access, such as SSL.
<code>java.naming.security.principal</code>	A string	none	The name of the principal for authenticating the caller to the service. This is built from other properties described below.
<code>java.naming.security.credentials</code>	A credential type	none	The type of credential used by the authentication scheme. Some examples include hashed password, clear-text password, key, or certificate,. If this property is unspecified, the behavior is determined by the service provider.
<code>principalDNprefix</code>	A string	none	Prefix added to the username to form the user DN. You can prompt the user for a username and build the fully-qualified DN by using the <code>principalDNprefix</code> and <code>principalDNSuffix</code> .
<code>principalDNSuffix</code>	string		Suffix added to the username to form the user DN. You can prompt the user for a username and build the fully-qualified DN by using the <code>principalDNprefix</code> and <code>principalDNSuffix</code> .
<code>useObjectCredential</code>	true or false	false	Whether the credential should be obtained as an opaque Object using the <code>org.jboss.security.auth.callback.ObjectCallback</code> type of Callback rather than as a <code>char[]</code> password using a JAAS

			PasswordCallback. This allows for passing non-char[] credential information to the LDAP server.
rolesCtxDN	A fully-qualified DN	none	The fully-qualified DN for the context to search for user roles.
userRolesCtxDNAttribute	An attribute name	none	The attribute in the user object that contains the DN for the context to search for user roles. This differs from rolesCtxDN in that the context to search for a user's roles may be unique for each user.
rolesAttributeID	An attribute	roles	Name of the attribute containing the user roles.
rolesAttributeIsDN	true or false	false	Whether or not the roleAttributeID contains the fully-qualified DN of a role object. If false, the role name is taken from the value of the roleNameAttributeId attribute of the context name. Certain directory schemas, such as Microsoft Active Directory, require this attribute to be set to true .
rolesNameAttributeID	An attribute	group	Name of the attribute within the roleCtxDN context which contains the role name. If the roleAttributeIsDN property is set to true , this property is used to find the role object's name attribute.
uidAttributeName	An attribute	uid	Name of the attribute in the UserRolesAttributeDN that corresponds to the user ID. This is used to locate the user roles.
matchOnUserDN	true or false	false	Whether or not the search for user roles should match on the user's fully distinguished DN or the username only. If true , the full user DN is used as the match value. If false , only the username is used as the match value against the uidAttributeName .

allowEmptyPasswords	true or false	true	Whether to allow empty passwords. Most LDAP servers treat empty passwords as anonymous login attempts. To reject empty passwords, set this to false.
----------------------------	---------------	------	--

Table 10.17. LdapExtended

Code	LdapExtended
Class	org.jboss.security.auth.spi.LdapExtLoginModule
Description	<p>An alternate LDAP login module implementation that uses searches to locate the bind user and associated roles. The roles query recursively follows DNs to navigate a hierarchical role structure. It uses the same java.naming options as the Ldap module, and uses the following options instead of the other options of the Ldap module.</p> <p>The authentication happens in 2 steps:</p> <ol style="list-style-type: none"> 1. An initial bind to the LDAP server is done using the bindDN and bindCredential options. The bindDN is a LDAP user with the ability to search both the baseCtxDN and rolesCtxDN trees for the user and roles. The user DN to authenticate against is queried using the filter specified by the baseFilter attribute. 2. The resulting user DN is authenticated by binding to the LDAP server using the user DN as the InitialLdapContext environment Context.SECURITY_PRINCIPAL. The Context.SECURITY_CREDENTIALS property is set to the String password obtained by the callback handler.

Table 10.18. LdapExtended Module Options

Option	Type	Default	Description
baseCtxDN	A fully-qualified DN	none	The fixed DN of the top-level context to begin the user search.
bindDN	A fully-qualified DN	none	The DN used to bind against the LDAP server for the user and roles queries. This DN needs read and search permissions on the baseCtxDN and rolesCtxDN values.
bindCredential	A string, optionally encrypted	none	The password for the bindDN . This can be encrypted if the jaasSecurityDomain is specified.
jaasSecurityDomain	A JMX ObjectName	none	The JMX ObjectName of the JaasSecurityDomain to use to decrypt the bindCredential . The encrypted form of the password is the format returned by the JaasSecurityDomain.encrypt64(byte[]) method.
baseFilter	LDAP filter string	none	A search filter used to locate the context of the user to authenticate. The input username or userDN obtained from the login module callback is substituted into the filter anywhere a {0} expression is used. A common example for the search filter is (uid={0}) .
rolesCtxDN	fully-qualified DN	none	The fixed DN of the context to search for user roles. This is not the DN where the actual roles are, but the DN where the objects containing the user roles are. For example, in a Microsoft Active Directory server, this is the DN where the user account is.
roleFilter	LDAP filter string		A search filter used to locate the roles associated with the authenticated user. The input username or userDN obtained from the login module callback is substituted into the filter anywhere a {0} expression is

			used. The authenticated userDN is substituted into the filter anywhere a {1} is used. An example search filter that matches on the input username is (member={0}) . An alternative that matches on the authenticated userDN is (member={1}) .
roleAttributeIsDN	true or false	false	Whether or not the roleAttributeID contains the fully-qualified DN of a role object. If false, the role name is taken from the value of the roleNameAttributeId attribute of the context name. Certain directory schemas, such as Microsoft Active Directory, require this attribute to be set to true .
defaultRole	Role name	none	A role included for all authenticated users
parseRoleNameFromDN	true or false	false	A flag indicating if the DN returned by a query contains the roleNameAttributeID. If set to true , the DN is checked for the roleNameAttributeID. If set to false , the DN is not checked for the roleNameAttributeID. This flag can improve the performance of LDAP queries.
parseUsername	true or false	false	A flag indicating if the DN is to be parsed for the username. If set to true , the DN is parsed for the username. If set to false the DN is not parsed for the username. This option is used together with usernameBeginString and usernameEndString .
usernameBeginString	a string	none	Defines the string which is to be removed from the start of the DN to reveal the username. This option is used together with usernameEndString .
usernameEndString	a string	none	Defines the string which is to be removed

			from the end of the DN to reveal the username. This option is used together with usernameBeginString .
roleNameAttributeID	An attribute	group	Name of the attribute within the roleCtxDN context which contains the role name. If the roleAttributeIsDN property is set to true , this property is used to find the role object's name attribute.
distinguishedNameAttribute	An attribute	distinguishedName	The name of the attribute in the user entry that contains the DN of the user. This may be necessary if the DN of the user itself contains special characters (backslash for example) that prevent correct user mapping. If the attribute does not exist, the entry's DN is used.
roleRecursion	An integer	0	The numbers of levels of recursion the role search will go below a matching context. Disable recursion by setting this to 0 .
searchTimeLimit	An integer	10000 (10 seconds)	The timeout in milliseconds for user or role searches.
searchScope	One of: OBJECT_SCOPE , ONELEVEL_SCOPE , SUBTREE_SCOPE	SUBTREE_SCOPE	The search scope to use.
allowEmptyPasswords	true or false	true	Whether to allow empty passwords. Most LDAP servers treat empty passwords as anonymous login attempts. To reject empty passwords, set this to false.

Table 10.19. RoleMapping

Code	RoleMapping
Class	org.jboss.security.auth.spi.RoleMappingLoginModule
Description	Maps a role which is the end result of the authentication process to a declarative role. This module must be flagged as optional when you add it to the security domain.

Table 10.20. RoleMapping Module Options

Option	Type	Default	Description
rolesProperties	The fully-qualified file path and name of a properties file or resource	roles.properties	The fully-qualified file path and name of a properties file or resource which maps roles to replacement roles. The format is original_role=role1,role2,role3
replaceRole	true or false	false	Whether to add to the current roles, or replace the current roles with the mapped ones. Replaces if set to true .

Table 10.21. RunAs

Code	RunAs
Class	class: org.jboss.security.auth.spi.RunAsLoginModule
Description	A helper module that pushes a run as role onto the stack for the duration of the login phase of authentication, and pops the run as role off the stack in either the commit or abort phase. This login module provides a role for other login modules that must access secured resources in order to perform their authentication, such as a login module which accesses a secured EJB. RunAsLoginModule must be configured before the login modules that require a run as role to be established.

Table 10.22. RunAs Options

Option	Type	Default	Description
roleName	A role name.	nobody	The name of the role to use as the run as role during the login phase.

Table 10.23. Simple

Code	simple
Class	org.jboss.security.auth.spi.SimpleServerLoginModule
Description	A module for quick setup of security for testing purposes. It implements the following simple algorithm: <ul style="list-style-type: none"> ▶ If the password is null, authenticate the user and assign an identity of guest and a role of guest. ▶ Otherwise, if the password is equal to the user, assign an identity equal to the username and both and and guest roles. ▶ Otherwise, authentication fails.

Simple Module Options

The **Simple** module has no options.

Table 10.24. ConfiguredIdentity

Code	ConfiguredIdentity
Class	org.picketbox.datasource.security.ConfiguredIdentityLoginModule
Description	Associates the principal specified in the module options with any subject authenticated against the module. The type of Principal class used is org.jboss.security.SimplePrincipal .

Table 10.25. ConfiguredIdentity Module Options

Option	Type	Default	Description
principal	Name of a principal.	guest	The principal which will be associated with any subject authenticated against the module.

Table 10.26. SecureIdentity

Code	SecureIdentity
Class	org.picketbox.datasource.security.SecureIdentityLoginModule
Description	This module is provided for legacy purposes. It allows you to encrypt a password and then use the encrypted password with a static principal. If your application uses SecureIdentity , consider using a password vault mechanism instead.

Table 10.27. SecureIdentity Module Options

Option	Type	Default	Description
username	string	none	The username for authentication.
password	encrypted string	none	The password to use for authentication. To encrypt the password, use the module directly at the command line. java org.picketbox.datasource.security.SecureIdentityLoginModule password_to_encrypt
managedConnectionFactoryName	A JCA resource	none	Paste the result of this command into the module option's value field.
managedConnectionFactoryName	A JCA resource	none	The name of the JCA connection factory for your datasource.

Table 10.28. PropertiesUsers

Code	PropertiesUsers
Class	org.jboss.security.auth.spi.PropertiesUsersLoginModule
Description	Uses a properties file to store usernames and passwords for authentication. No authorization (role mapping) is provided. This module is only appropriate for testing.

Table 10.29. PropertiesUsers Module Options

Option	Type	Default	Description
properties	The fully-qualified file path and name of a Java properties file or resource.	none	The properties file containing the usernames and clear-text passwords to be used for authentication.

Table 10.30. SimpleUsers

Code	SimpleUsers
Class	org.jboss.security.auth.spi.SimpleUsersLoginModule
Description	This login module stores the username and clear-text password in a Java properties file. It is included for testing only, and is not appropriate for a production environment.

Table 10.31. SimpleUsers Module Options

Option	Type	Default	Description
username	string	none	The username to use for authentication.
password	string	none	The clear-text password to use for authentication.

Table 10.32. LdapUsers

Code	LdapUsers
Class	org.jboss.security.auth.spi.LdapUsersLoginModule
Description	The LdapUsers module is superseded by the ExtendedLDAP and AdvancedLdap modules.

Table 10.33. Kerberos

Code	Kerberos
Class	com.sun.security.auth.module.Krb5LoginModule
Description	Performs Kerberos login authentication, using GSSAPI. This module is part of the security framework from the API provided by Sun Microsystems. Details can be found at http://docs.oracle.com/javase/1.4.2/docs/guide/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html . This module needs to be paired with another module which handles the authentication and roles mapping.

Table 10.34. Kerberos Module Options

Option	Type	Default	Description
storekey	true or false	false	Whether or not to add the KerberosKey to the subject's private credentials.
doNotPrompt	true or false	false	If set to true , the user is not prompted for the password.
useTicketCache	Boolean value of true or false .	false	If true , the GTG is obtained from the ticket cache. If false , the ticket cache is not used.
ticketcache	A file or resource representing a Kerberos ticket cache.	The default depends on which operating system you use.	The location of the ticket cache. <ul style="list-style-type: none"> ▶ Red Hat Enterprise Linux / Solaris: /tmp/krb5cc_uid, using the numeric UID value of the operating system. ▶ Microsoft Windows Server: uses the Local Security Authority (LSA) API to find the ticketcache.
useKeyTab	true or false	false	Whether to obtain the principal's key from a key table file.
keytab	A file or resource representing a Kerberos keytab.	the location in the operating system's Kerberos configuration file, or /home/user/krb5.keytab	The location of the key table file.
principal	A string	none	The name of the principal. This can either be a simple user name or a service name such as host/testserver.acme.com . Use this instead of obtaining the principal from the key table, or when the key table contains more than one principal.
useFirstPass	true or false	false	Whether to retrieve the username and password from the module's shared state, using javax.security.auth.login.name and javax.security.auth.login.password as the keys. If authentication fails, no retry attempt is made.

tryFirstPass	true or false	false	Same as useFirstPass , but if authentication fails, the module uses the CallbackHandler to retrieve a new username and password. If the second authentication fails, the failure is reported to the calling application.
storePass	true or false	false	Whether to store the username and password in the module's shared state. This does not happen if the keys already exist in the shared state, or if authentication fails.
clearPass	true or false	false	Set this to true to clear a the username and password from the shared state after both phases of authentication complete.

Table 10.35. SPNEGOUsers

Code	SPNEGOUsers
Class	org.jboss.security.negotiation.spnego.SPNEGOLoginModule
Description	Allows SPNEGO authentication to a Microsoft Active Directory server or other environment which supports SPNEGO. SPNEGO can also carry Kerberos credentials. This module needs to be paired with another module which handles authentication and role mapping.

Table 10.36. SPNEGO Module Options

Option	Type	Default	Description
storeKey	true or false	false	Whether or not to store the key.
useKeyTab	true or false	false	Whether to use a key table.
principal	String reperesenting a principal for Kerberos auhtentication.	none	The name of the principal for authentication.
keyTab	A file or resource representing a keytab.	none	The location of a key table.
doNotPrompt	true or false	false	Whether to prompt for a password.
debug	true or false	false	Whether to record more verbose messages for debugging purposes.

Table 10.37. AdvancedLdap

Code	AdvancedLdap
Class	org.jboss.security.negotiation.AdvancedLdapLoginModule
Description	A module which provides additional functionality, such as SASL and the use of a JAAS security domain.

Table 10.38. AdvancedLdap Module Options

Option	Type	Default	Description
bindAuthentication	string	none	The type of SASL authentication to use for binding to the directory server.
jassSecurityDomain	string	none	The name of the JAAS security domain to use.
java.naming.provider.url	string	none	The URI of the directory server.
baseCtxDN	A fully qualified Distinguished Name (DN).	none	The distinguished name to use as the base for searches.
baseFilter	String representing a LDAP search filter.	none	The filter to use to narrow down search results.
roleAttributeID	A string representing an LDAP attribute.	none	The LDAP attribute which contains the names of authorization roles.
roleAttributeIsDN	true or false	false	Whether the role attribute is a Distinguished Name (DN).
roleNameAttributeID	String representing an LDAP attribute.	none	The attribute contained within the RoleAttributeId which contains the actual role attribute.
recurseRoles	true or false	false	Whether to recursively search the RoleAttributeId for roles.

Table 10.39. AdvancedADLdap

Code	AdvancedADLdap
Class	org.jboss.security.negotiation.AdvancedADLoginModule
Description	This module extends the AdvancedLdap login module, and adds extra parameters that are relevant to Microsoft Active Directory.

Table 10.40. AdvancedADLdap Module Options

Option	Type	Default	Description
primaryGroupID	A string representing a Microsoft Active Directory group ID.	none	A primary group for limiting authorization. The primary group ID is used to obtain the objectSid of the user, for authorization.

Table 10.41. UsersRoles

Code	UsersRoles
Class	org.jboss.security.auth.spi.UsersRoleLoginModule
Description	A simple login module that supports multiple users and user roles stored in two different properties files.

Table 10.42. UsersRoles Module Options

Option	Type	Default	Description
usersProperties	Path to a file or resource.	users.properties	The file or resource which contains the user-to-password mappings. The format of the file is user=hashed-password
rolesProperties	Path to a file or resource.	roles.properties	The file or resource which contains the user-to-role mappings. The format of the file is username=role1,role2,role3
password-stacking	useFirstPass or false	false	A value of useFirstPass indicates that this login module should first look to the information stored in the LoginContext for the identity. This option can be used when stacking other login modules with this one.
hashAlgorithm	A string representing a password hashing algorithm.	none	The name of the java.security.MessageDigest algorithm to use to hash the password. There is no default so this option must be explicitly set to enable hashing. When hashAlgorithm is specified, the clear text password obtained from the CallbackHandler is hashed before it is passed to UsernamePasswordLoginModule.validatePassword as the inputPassword argument. The password stored in the users.properties file must be comparably hashed.
hashEncoding	base64 or hex	base64	The string format for the hashed password, if hashAlgorithm is also set.
hashCharset	A string	The default encoding set in the container's runtime environment	The encoding used to convert the clear-text password to a byte array.
unauthenticatedIdentity	A principal name	none	Defines the principal name assigned to requests which contain no authentication information. This can allow unprotected

servlets to invoke methods on EJBs that do not require a specific role. Such a principal has no associated roles and can only access unsecured EJBs or EJB methods that are associated with the **unchecked permission** constraint.

Custom Authentication Modules

Authentication modules are implementations of `org.jboss.security.LoginModule`. Refer to the API documentation for more information about creating a custom authentication module.

[Report a bug](#)

10.2. Included Authorization Modules

The following modules provide authorization services.

Code	Class
DenyAll	org.jboss.security.authorization.modules.AllDenyAuthorizationModule
PermitAll	org.jboss.security.authorization.modules.AllPermitAuthorizationModule
Delegating	org.jboss.security.authorization.modules.DelegatingAuthorizationModule
Web	org.jboss.security.authorization.modules.WebAuthorizationModule
JACC	org.jboss.security.authorization.modules.JACCAuthorizationModule

[Report a bug](#)

10.3. Included Security Mapping Modules

The following security mapping roles are provided in the JBoss Enterprise Application Platform.

Code	Class
PropertiesRoles	org.jboss.security.mapping.providers.role.PropertiesRolesMappingProvider
SimpleRoles	org.jboss.security.mapping.providers.role.SimpleRolesMappingProvider
DeploymentRoles	org.jboss.security.mapping.providers.DeploymentRolesMappingProvider
DatabaseRoles	org.jboss.security.mapping.providers.role.DatabaseRolesMappingProvider
LdapRoles	org.jboss.security.mapping.providers.role.LdapRolesMappingProvider

[Report a bug](#)

10.4. Included Security Auditing Provider Modules

The JBoss Enterprise Application Platform provides one security auditing provider.

Code	Class

LogAuditProvider	org.jboss.security.audit.providers.LogAuditProvider
------------------	---

[Report a bug](#)

Chapter 11. The Logging Subsystem

11.1. Introduction

11.1.1. Overview of Logging

JBoss Enterprise Application Platform 6 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 it supports several third party application logging frameworks in addition to JBoss Logging.

The logging subsystem is configured using a system of log categories and log handlers. Log categories define what messages to capture, and log handlers define how to deal with those messages (write to disk, send to console etc).

All of this configuration can be performed in the Management Console or by using the CLI tool.

[Report a bug](#)

11.1.2. Application Logging Frameworks Supported By JBoss LogManager

JBoss LogManager supports the following logging frameworks:

- » JBoss Logging - included with JBoss Enterprise Application Platform 6
- » Apache Commons Logging - <http://commons.apache.org/logging/>
- » Simple Logging Facade for Java (SLF4J) - <http://www.slf4j.org/>
- » Apache log4j - <http://logging.apache.org/log4j/1.2/>
- » Java SE Logging (java.util.logging) - <http://download.oracle.com/javase/6/docs/api/java/util/logging/package-summary.html>

[Report a bug](#)

11.1.3. Configure Boot Logging

The boot log is the record of events that occur while the server is starting up (or "booting").

The boot log can be configured by editing the **logging.properties** file. This file is a standard Java properties file and can be edited in a text editor. Each line in the file has the format of **property=value**.

Depending on whether you run the JBoss Enterprise Application Platform as a managed domain or standalone server, the full path to the **logging.properties** file is either **EAP_HOME/domain/configuration/logging.properties** or **EAP_HOME/standalone/configuration/logging.properties**.

[Report a bug](#)

11.1.4. Default Log File Locations

These are the log files that get created for the default logging configurations. The default configuration writes the server log files using periodic log handlers

Table 11.1. Default Log Files for a standalone server

Log File	Description
EAP_HOME/standalone/log/boot.log	The server boot log. Contains log messages related to the startup of the server.
EAP_HOME/standalone/log/server.log	The Server Log. Contains all log messages once the server has launched.

Table 11.2. Default Log Files for a managed domain

Log File	Description
<code>EAP_HOME/domain/log/host-controller/boot.log</code>	Host Controller boot log. Contains log messages related to the startup of the host controller.
<code>EAP_HOME/domain/log/process-controller/boot.log</code>	Process controller boot log. Contains log messages related to the startup of the process controller.
<code>EAP_HOME/domain/servers/SERVERNAME/logs/boot.log</code>	Server Boot log for the named server. Contains log messages related to the startup of the specified server.
<code>EAP_HOME/domain/servers/SERVERNAME/logs/server.log</code>	The server log for the named server. Contains all log messages for that server once it has launched.

[Report a bug](#)

11.1.5. About Log Levels

Log levels are an ordered set of enumerated values that indicate the nature and severity of a log message. The level of a given log message is specified by the developer using the appropriate methods of their chosen logging framework to send the message.

JBoss Enterprise Application Platform 6 supports all the log levels used by the supported application logging frameworks. The most commonly used six log levels are (in order of lowest to highest): **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 which 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**.

[Report a bug](#)

11.1.6. Supported Log Levels

Table 11.3. Supported Log Levels

Log Level	Value	Description
FINEST	300	-
FINER	400	-
TRACE	400	Use for messages that provide detailed information about the running state of an application. Log messages of TRACE are usually only captured when debugging an application.
DEBUG	500	Use for messages that indicate the progress individual requests or activities of an application. Log messages of DEBUG are usually only captured when debugging an application.
FINE	500	-
CONFIG	700	-
INFO	800	Use for messages that indicate the overall progress of the application. Often used for application startup, shutdown and other major lifecycle events.
WARN	900	Use to indicate a situation that is not in error but is not considered ideal. May indicate circumstances that may lead to errors in the future.
WARNING	900	-
ERROR	1000	Use to indicate an error that has occurred that could prevent the current activity or request from completing but will not prevent the application from running.
FATAL	1100	Use to indicate events that could cause critical service failure and application shutdown and possibly cause JBoss Enterprise Application Platform 6 to shutdown.

[Report a bug](#)

11.1.7. About Log Categories

Log categories define a set of log messages to capture and one or more log handlers which will process the messages.

The log messages to capture are defined by their Java package of origin and log level. Messages from classes in that package and of that log level or lower are captured by the log category and sent to the specified log handlers.

Log categories can optionally use the log handlers of the root logger instead of their own handlers.

[Report a bug](#)

11.1.8. About the Root Logger

The root logger captures all log messages sent to the server (of a specified level) that are not captured by a log category. These messages are then sent to one or more log handlers.

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 file **server.log**. This file is sometimes referred to as the server log.

[Report a bug](#)

11.1.9. About Log Handlers

Log handlers define how captured log messages are recorded by JBoss Enterprise Application Platform. There are six types of log handler that can be configured: **Console**, **File**, **Periodic**, **Size**, **Async** and **Custom**.

[Report a bug](#)

11.1.10. Types of Log Handlers

Console

Console log handlers write log messages to either the host operating system's standard out (stdout) or standard error (stderr) stream. These messages are displayed when JBoss Enterprise Application Platform 6 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

File log handlers are the simplest log handlers that write log messages to a specified file.

Periodic

Periodic file handlers write log messages to a named file until a specified period of time has elapsed. Once the time period has past then 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

Size log handlers write 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 prefix 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.

Async

Async log handlers are wrapper log handlers that provide asynchronous behaviour for one or more other log handlers. These are useful for log handlers that may have high latency or other performance problems such as writing a log file to a network file system.

Custom

Custom log handlers enable to 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.

[Report a bug](#)

11.1.11. About Log Formatters

A log formatter is the configuration property of a log handler that defines the appearance of log messages from that handler. It is a string that uses a syntax based on `java.util.Formatter` class.

For example the log formatter string from the default configuration, `%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n`, creates log messages that look like:

```
15:53:26,546 INFO [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin
console listening on http://127.0.0.1:9990
```

[Report a bug](#)

11.1.12. Log Formatter Syntax

Table 11.4. Log Formatter Syntax

Symbol	Description
<code>%c</code>	The category of the logging event
<code>%p</code>	The level of the log entry (info/debug/etc)
<code>%P</code>	The localized level of the log entry
<code>%d</code>	The current date/time (yyyy-MM-dd HH:mm:ss,SSS form)
<code>%r</code>	The relative time (milliseconds since the log was initialised)
<code>%z</code>	The time zone
<code>%k</code>	A log resource key (used for localization of log messages)
<code>%m</code>	The log message (including exception trace)
<code>%s</code>	The simple log message (no exception trace)
<code>%e</code>	The exception stack trace (no extended module information)
<code>%E</code>	The exception stack trace (with extended module information)
<code>%t</code>	The name of the current thread
<code>%n</code>	A newline character
<code>%C</code>	The class of the code calling the log method (slow)
<code>%F</code>	The filename of the class calling the log method (slow)
<code>%l</code>	The source location of the code calling the log method (slow)
<code>%L</code>	The line number of the code calling the log method (slow)
<code>%M</code>	The method of the code calling the log method (slow)
<code>%x</code>	The Log4J Nested Diagnostic Context
<code>%X</code>	The Log4J Message Diagnostic Context
<code>%%</code>	A literal percent character (escaping)

[Report a bug](#)

11.2. Configure Logging in the Management Console

The management console provides a graphical user interface for the configuration of the root logger, log handlers, and log categories. You can find the logger configuration in the management console by following these steps:

You can access this configuration by following steps below:

1. Login to the Management Console
2. Navigate to the logging subsystem configuration. This step varies between servers running as standalone servers and servers running in a managed domain.
 - A. Standalone Server

Click on Profile, expand Core in the Profile pane, and then click on Logging.

B. Managed Domain

Click on Profile, select the profile to edit, expand Core, and then click on Logging.

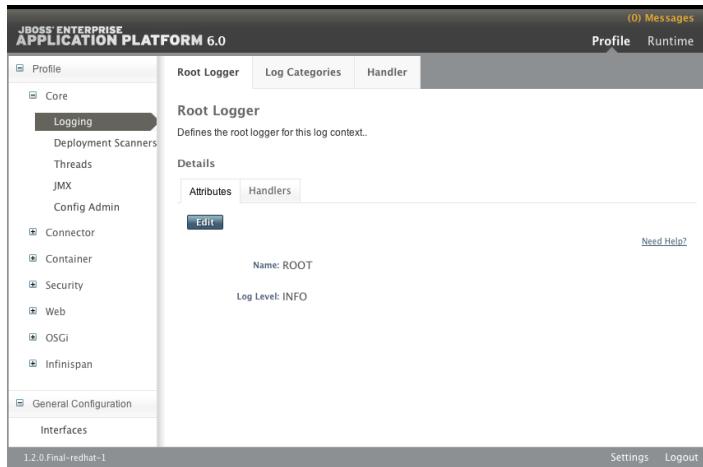


Figure 11.1. Logging Configuration in the Management Console

The tasks you can perform to configure the root logger are:

- ▶ Edit the log level.
- ▶ Add and remove log handlers.

The tasks you can perform to configure log categories are:

- ▶ Add and remove log categories.
- ▶ Edit log category properties.
- ▶ Add and remove log handlers from a category.

The main tasks you can perform to configure log handlers are:

- ▶ Adding new handlers.
- ▶ Configuring handlers.

All six supported log handlers (including custom) can be configured in the management console.

[Report a bug](#)

11.3. Logging Configuration in the CLI

11.3.1. Configure the Root Logger with the CLI

The root logger configuration can be viewed and edited using the CLI.

The main tasks you will perform to configuration the root logger are:

- ▶ Display the root logger configuration.
- ▶ Change the log level.
- ▶ Add log handlers to the root logger.
- ▶ Remove log handlers from the root logger.

Display the Contents of the Root Logger Configuration

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/root-logger=ROOT:read-resource
```

Example 11.1. Root Logger read-resource operation

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:read-resource
{
  "outcome" => "success",
  "result" => {
    "filter" => {"match" => "names"},
    "handlers" => [
      "CONSOLE",
      "FILE"
    ],
    "level" => "INFO"
  }
}
```

Set the Log Level of the Root Logger

Use the **write-attribute** operation with the following syntax where **LEVEL** is one of the supported log levels.

```
/subsystem=logging/root-logger=ROOT:write-attribute(name="level",
value="LEVEL")
```

Example 11.2. Root Logger write-attribute operation to set the log level

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Add a Log Handler to the Root Logger

Use the **root-logger-assign-handler** operation with the following syntax where **HANDLER** is the name of the log handler to be added.

```
/subsystem=logging/root-logger=ROOT:root-logger-assign-handler(name="HANDLER")
```

The log handler must already have been created before it can be added to the root logger.

Example 11.3. Root Logger root-logger-assign-handler operation

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:root-logger-assign-handler(name="AccountsNFSAsync")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove a Log Handler from the Root Logger

Use the **root-logger-unassign-handler** with the following syntax, where **HANDLER** is the name of the log handler to be removed.

```
/subsystem=logging/root-logger=ROOT:root-logger-unassign-handler(name="HANDLER")
```

Example 11.4. Remove a Log Handler

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:root-logger-unassign-handler(name="AccountsNFSAsync")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

11.3.2. Configure a Log Category in the CLI

Log categories can be added, removed and edited in the CLI.

The main tasks you will perform to configure a log category are:

- ▶ Display the configuration of a log category.
- ▶ Add a new log category.
- ▶ Set the log level.
- ▶ Add log handlers to a log category.
- ▶ Remove log handlers from a log category.
- ▶ Remove a log category.

Display a log category configuration

Use the **read-resource** operation with the following syntax. Replace **CATEGORY** with the name of the category.

```
/subsystem=logging/logger= CATEGORY:read-resource
```

Example 11.5. Log Category read-resource operation

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=org.apache.tomcat.util.modeler:read-resource
{
    "outcome" => "success",
    "result" => {
        "filter" => undefined,
        "handlers" => undefined,
        "level" => "WARN",
        "use-parent-handlers" => true
    }
}
[standalone@localhost:9999 /]
```

Add a log category

Use the **add** operation with the following syntax. Replace **CATEGORY** with the category to be added.

```
/subsystem=logging/logger= CATEGORY:add
```

Example 11.6. Adding a new log category

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:add
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the log level

Use the **write-attribute** operation with the following syntax. Replace **CATEGORY** with the name of the log category and **LEVEL** with the log level that is to be set.

```
/subsystem=logging/logger= CATEGORY:write-attribute(name="level",
value=" LEVEL")
```

Example 11.7. Setting a log level

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:write-
attribute(name="level", value="DEBUG")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the log category to use the log handlers of the root logger.

Use the **write-attribute** operation with the following syntax. Replace **CATEGORY** with the name of the log category. Replace **BOOLEAN** with true for this log category to use the handlers of the root logger. Replace it with false if it is to use only its own assigned handlers.

```
/subsystem=logging/logger=CATEGORY:write-attribute(name="use-parent-
handlers", value="BOOLEAN")
```

Example 11.8. Setting use-parent-handlers

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:write-
attribute(name="use-parent-handlers", value="true")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Add a log handlers to a log category

Use the **assign-handler** operation with the following syntax. Replace **CATEGORY** with the name of the category and **HANDLER** with the name of the handler to be added.

```
/subsystem=logging/logger=CATEGORY:assign-handler(name="HANDLER")
```

The log handler must already have been created before it can be added to the root logger.

Example 11.9. Adding a log handler

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:assign-
handler(name="AccountsNFSAsync")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove a log handler from a log category

Use the **unassign-handler** operation with the following syntax. Replace **CATEGORY** with the name of the category and **HANDLER** with the name of the log handler to be removed.

```
/subsystem=logging/logger=CATEGORY:unassign-handler(name="HANDLER")
```

Example 11.10. Removing a log handler

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:root-
logger-unassign-handler(name="AccountsNFSAsync")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove a category

Use the **remove** operation with the following syntax. Replace **CATEGORY** with the name of the category to be removed.

```
/subsystem=logging/logger=CATEGORY:remove
```

Example 11.11. Removing a log category

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)**11.3.3. Configure a Console Log Handler in the CLI**

Console log handlers can be added, removed and edited in the CLI.

Procedure 11.1. Display a console log handler configuration

- ▶ Use the **read-resource** operation with the following syntax. Replace **HANDLER** with the name of the console log handler.

```
/subsystem=logging/console-handler=HANDLER:read-resource
```

Example 11.12.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=CONSOLE:read-resource
{
    "outcome" => "success",
    "result" => {
        "autoflush" => true,
        "encoding" => undefined,
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "INFO",
        "target" => "System.out"
    }
}
[standalone@localhost:9999 /]
```

Procedure 11.2. Add a Console Log Handler

- ▶ Use the **add** operation with the following syntax. Replace **HANDLER** with the console log handler to be added.

```
/subsystem=logging/console-handler=HANDLER:add
```

Example 11.13.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:add
>{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Procedure 11.3. Set the Log Level

- ▶ Use the **change-log-level** operation with the following syntax. Replace **HANDLER** with the name of the console log handler and **LEVEL** with the log level that is to be set.

```
/subsystem=logging/console-handler=HANDLER:change-log-level(level="LEVEL")
```

Example 11.14.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:change-log-level(level="TRACE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Procedure 11.4. Set the Target

- Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the console log handler. Replace **TARGET** with either **System.out** or **System.err** for the system error stream or standard out stream respectively.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="target",
value="TARGET")
```

Example 11.15.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:write-attribute(name="target", value="System.out")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Procedure 11.5. Set the Encoding

- Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the console log handler. Replace **ENCODING** with the name of the required character encoding system.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

Example 11.16.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:write-attribute(name="encoding", value="utf-8")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Procedure 11.6. Set the Formatter

- Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the console log handler. Replace **FORMAT** with the required formatter string.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="formatter",
value="FORMAT")
```

Example 11.17.

```
[standalone@1/subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"))
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Procedure 11.7. Set the Auto Flush

- Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the console log handler. Replace **BOOLEAN** with **true** if this handler is to immediately write its output.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

Example 11.18.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Procedure 11.8. Remove a Console Log Handler

- ▶ Use the **remove** operation with the following syntax. Replace **HANDLER** with the name of the console log handler to be removed.

```
/subsystem=logging/console-handler=HANDLER:remove
```

Example 11.19.

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

11.3.4. Configure a File Log Handler in the CLI

File log handlers can be added, removed and edited in the CLI.

The main tasks you will perform to configure a file log handler are:

- ▶ Display the configuration of a file log handler
- ▶ Add a new file log handler.
- ▶ Set the handler's log level.
- ▶ Set the handler's appending behaviour.
- ▶ Set whether the handler uses autoflush or not.
- ▶ Set the encoding used for the handler's output.
- ▶ Specify the file to which the log handler will write.
- ▶ Set the formatter used for the handler's output.
- ▶ Remove a file log handler.

Display a file log handler configuration

Use the **read-resource** operation with the following syntax. Replace **HANDLER** with the name of the file log handler.

```
/subsystem=logging/file-handler=HANDLER:read-resource
```

Example 11.20. Using the read-resource operation

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "accounts.log",
            "relative-to" => "jboss.server.log.dir"
        },
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => undefined
    }
}
[standalone@localhost:9999 /]
```

Add a file log handler

Use the **add** operation with the following syntax. Replace **PATH** with the filename for the file that the log is being written to. Replace **DIR** with the name of the directory where the file is to be located. The value of **DIR** can be a path variable.

```
/subsystem=logging/file-handler=HANDLER:add(file={"path"=>"PATH", "relative-to"=>"DIR"})
```

Example 11.21. Add a file log handler

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:add(file={"path"=>"accounts.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the Log level

Use the **change-log-level** operation with the following syntax. Replace **HANDLER** with the name of the file log handler. Replace **LEVEL** with the log level that is to be set.

```
/subsystem=logging/file-handler=HANDLER:change-log-level(level="LEVEL")
```

Example 11.22. Changing the log level

```
/subsystem=logging/file-handler=accounts_log:change-log-
level(level="DEBUG"){"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the append behaviour

Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the file log handler. Replace **BOOLEAN** with **false** if you required that a new log file be created each time the application server is launched. Replace **BOOLEAN** with **true** if the application server should continue to use the same file.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="append", value="BOOLEAN")
```

Example 11.23. Changing the append property

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /]
```

JBoss Enterprise Application Platform 6 must be restarted for this change to take effect.

Set the Auto Flush

Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the file log handler. Replace **BOOLEAN** with **true** if this handler is to immediately write its output.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

Example 11.24. Changing the autoflush property

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="autoflush", value="false")
{
    "outcome" => "success",
    "response-headers" => {"process-state" => "reload-required"}
}
[standalone@localhost:9999 /]
```

JBoss Enterprise Application Platform 6 must be restarted for this change to take effect.

Set the Encoding

Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the file log handler. Replace **ENCODING** with the name of the required character encoding system.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

Example 11.25. Set the Encoding

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Change the file to which the log handler writes

Use the **change-file** operation with the following syntax. Replace **PATH** with the filename for the file that the log is being written to. Replace **DIR** with the name of the directory where the file is to be located. The value of **DIR** can be a path variable.

```
/subsystem=logging/file-handler=HANDLER:change-file(file={"path"=>"PATH",
"relative-to"=>"DIR"})
```

Example 11.26. Change the file to which the log handler writes

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:change-file(file={"path">"accounts-debug.log",
"relative-to"=>"jboss.server.log.dir"})
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the Formatter

Use the **write-attribute** operation with the following syntax. Replace **HANDLER** with the name of the console log handler. Replace **FORMAT** with the required formatter string.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="formatter",
value="FORMAT")
```

Example 11.27. Set the Formatter

```
[standalone@1/subsystem=logging/file-handler=accounts-log:write-
attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t)
%s%E%n"))
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove a File Log Handler

Use the **remove** operation with the following syntax. Replace **HANDLER** with the name of the file log handler to be removed.

```
/subsystem=logging/file-handler=HANDLER:remove
```

Example 11.28. Remove a Console Log Handler

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:remove
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

A log handler can only be removed if it is not being referenced by a log category or an async log handler.

[Report a bug](#)

11.3.5. Configure a Periodic Log Handler in the CLI

Periodic log handlers can be added, removed and edited in the CLI.

The main tasks you will perform to configure a periodic log handler are:

- ▶ Display the configuration of a periodic log handler
- ▶ Add a new periodic log handler.
- ▶ Set the handler's log level.
- ▶ Set the handler's appending behaviour.
- ▶ Set whether the handler uses autoflush or not.
- ▶ Set the encoding used for the handler's output.
- ▶ Specify the file to which the log handler will write.
- ▶ Set the formatter used for the handler's output.
- ▶ Set the suffix for rotated logs
- ▶ Remove a periodic log handler.

Each of those tasks are described below.

Display a Periodic Rotating File log handler configuration

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:read-resource
```

Replace **HANDLER** with the name of the file log handler.

Example 11.29. Using the read-resource operation

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "daily-debug.log",
            "relative-to" => "jboss.server.log.dir"
        },
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => undefined
    }
}
[standalone@localhost:9999 /]
```

Add a new Periodic Rotating File log handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-
handler=HANDLER:add(file={"path"=>"PATH", "relative-to"=>"DIR"},
suffix=SUFFIX)
```

Replace **HANDLER** with the name of the log handler. Replace **PATH** with the filename for the file that the log is being written to. Replace **DIR** with the name of the directory where the file is to be located. The value of **DIR** can be a path variable. Replace **SUFFIX** with the file rotation suffix to be used.

Example 11.30. Add a new handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:add(file={"path"=>"daily-debug.log", "relative-
to"=>"jboss.server.log.dir"}, suffix=".yyyy.MM.dd")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the Log level

Use the **change-log-level** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:change-log-
level(level="LEVEL")
```

Replace **HANDLER** with the name of the periodic log handler. Replace **LEVEL** with the log level that is to be set.

Example 11.31. Set the log level

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:change-log-level(level="DEBUG")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the append behaviour

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-handler=HANDLER:write-
attribute(name="append", value="BOOLEAN")
```

Replace **HANDLER** with the name of the periodic log handler. Replace **BOOLEAN** with **false** if you required that a new log file be created each time the application server is launched. Replace **BOOLEAN** with **true** if the application server should continue to use the same file.

JBoss Enterprise Application Platform 6 must be restarted for this change to take effect.

Example 11.32. Set the append behaviour

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /]
```

Set the Auto Flush

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")
```

Replace **HANDLER** with the name of the periodic log handler. Replace **BOOLEAN** with **true** if this handler is to immediately write its output.

JBoss Enterprise Application Platform 6 must be restarted for this change to take effect.

Example 11.33. Set the Auto Flush behaviour

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="autoflush", value="false")
{
    "outcome" => "success",
    "response-headers" => {"process-state" => "reload-required"}
}
[standalone@localhost:9999 /]
```

Set the Encoding

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-
attribute(name="encoding", value="ENCODING")
```

Replace **HANDLER** with the name of the periodic log handler. Replace **ENCODING** with the name of the required character encoding system.

Example 11.34. Set the Encoding

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Change the file to which the log handler writes

Use the **change-file** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handle=HANDLER:change-
file(file={"path"=>"PATH", "relative-to"=>"DIR"})
```

Replace **HANDLER** with the name of the periodic log handler. Replace **PATH** with the filename for the file that the log is being written to. Replace **DIR** with the name of the directory where the file is to be located. The value of **DIR** can be a path variable.

Example 11.35. Change the file to which the log handler writes

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handle=HOURLY_DEBUG:change-file(file={"path"=>"daily-debug.log",
"relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the Formatter

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handle=HANDLER:write-
attribute(name="formatter", value="FORMAT")
```

Replace **HANDLER** with the name of the periodic log handler. Replace **FORMAT** with the required formatter string.

Example 11.36. Set the Formatter

```
[standalone@l/subsystem=logging/periodic-rotating-file-
handle=HOURLY_DEBUG:write-attribute(name="formatter",
value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the suffix for rotated logs

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handle=HANDLER:write-
attribute(name="suffix", value="SUFFIX")
```

Replace **HANDLER** with the name of the console log handler. Replace **SUFFIX** with the required suffix string.

Example 11.37.

```
[standalone@l/subsystem=logging/periodic-rotating-file-
handle=HOURLY_DEBUG:write-attribute(name="suffix", value=".yyyy-MM-dd-
HH"))
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove a periodic log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:remove
```

Replace **HANDLER** with the name of the periodic log handler.

Example 11.38. Remove a periodic log handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)**11.3.6. Configure a Size Log Handler in the CLI**

Size rotated file log handlers can be added, removed and edited in the CLI.

The tasks you will perform to configure a size rotated file log handler are:

- ▶ Display the configuration of the log handler
- ▶ Add a new log handler.
- ▶ Set the handler's log level.
- ▶ Set the handler's appending behaviour.
- ▶ Set whether the handler uses autoflush or not.
- ▶ Set the encoding used for the handler's output.
- ▶ Specify the file to which the log handler will write.
- ▶ Set the formatter used for the handler's output.
- ▶ Set the maximum size of each log file
- ▶ Set the maximum number of backup logs to keep
- ▶ Remove a log handler.

Each of these tasks are described below.

Display the configuration of the log handler

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:read-resource
```

Replace **HANDLER** with the name of the log handler.

Example 11.39. Display the configuration of the log handler

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "accounts_trace.log",
            "relative-to" => "jboss.server.log.dir"
        },
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => undefined,
        "max-backup-index" => 1,
        "rotate-size" => "2m"
    }
}
[standalone@localhost:9999 /]
```

Add a new log handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-
handler=HANDLER:add(file={"path"=>"accounts_trace.log", "relative-
to"=>"jboss.server.log.dir"})
```

Replace **HANDLER** with the name of the log handler. Replace **PATH** with the filename for the file that the log is being written to. Replace **DIR** with the name of the directory where the file is to be located. The value of **DIR** can be a path variable.

Example 11.40. Add a new log handler

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:add(file={"path"=>"accounts_trace.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the handler's log level

Use the **change-log-level** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:change-log-
level(level="TRACE")
```

Replace **HANDLER** with the name of the log handler. Replace **LEVEL** with the log level that is to be set.

Example 11.41. Set the handler's log level

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:change-log-level(level="TRACE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the handler's appending behaviour

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="append", value="BOOLEAN")
```

Replace **HANDLER** with the name of the log handler. Replace **BOOLEAN** with **false** if you required that a new log file be created each time the application server is launched. Replace **BOOLEAN** with **true** if the application server should continue to use the same file.

JBoss Enterprise Application Platform 6 must be restarted for this change to take effect.

Example 11.42. Set the handler's appending behaviour

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /]
```

Set whether the handler uses autoflush or not

Use the **write-attribute** operation with the following syntax.

Replace **HANDLER** with the name of the log handler. Replace **BOOLEAN** with **true** if this handler is to immediately write its output.

Example 11.43. Set whether the handler uses autoflush or not

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the encoding used for the handler's output

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="encoding", value="ENCODING")
```

Replace **HANDLER** with the name of the log handler. Replace **ENCODING** with the name of the required character encoding system.

Example 11.44. Set the encoding used for the handler's output

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="encoding", value="utf-8")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Specify the file to which the log handler will write

Use the **change-file** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:change-
file(file={"path"=>"PATH", "relative-to"=>"DIR"})
```

Replace **HANDLER** with the name of the log handler. Replace **PATH** with the filename for the file that the log is being written to. Replace **DIR** with the name of the directory where the file is to be located. The value of **DIR** can be a path variable.

Example 11.45. Specify the file to which the log handler will write

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:change-file(file={"path"=>"accounts_trace.log",
 "relative-to"=>"jboss.server.log.dir"})
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the formatter used for the handler's output

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="formatter", value="FORMATTER")
```

Replace **HANDLER** with the name of the log handler. Replace **FORMAT** with the required formatter string.

Example 11.46. Set the formatter used for the handler's output

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="formatter",
value="%d{HH:mm:ss,SSS} %-5p (%c) [%t] %s%E%n")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the maximum size of each log file

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="rotate-size", value="SIZE")
```

Replace **HANDLER** with the name of the log handler. Replace **SIZE** with maximum file size.

Example 11.47. Set the maximum size of each log file

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="rotate-size", value="50m")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the maximum number of backup logs to keep

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="max-backup-index", value="NUMBER")
```

Replace **HANDLER** with the name of the log handler. Replace **NUMBER** with the required number of log files to keep.

Example 11.48. Set the maximum number of backup logs to keep

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="max-backup-index",
value="5")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove a log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:remove
```

Replace **HANDLER** with the name of the log handler.

Example 11.49. Remove a log handler

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

11.3.7. Configure a Async Log Handler in the CLI

Async log handlers can be added, removed and edited in the CLI.

The tasks you will perform to configure an async log handler are:

- ▶ Display the configuration of an async log handler
- ▶ Add a new async log handler
- ▶ Change the log level
- ▶ Set the queue length

- ▶ Set the overflow action
- ▶ Add sub-handlers
- ▶ Remove sub-handlers
- ▶ Remove an async log handler

Each of these tasks are described below.

Display the configuration of an async log handler

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:read-resource
```

Replace **HANDLER** with the name of the log handler.

Example 11.50.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:read-resource
{
    "outcome" => "success",
    "result" => {
        "encoding" => undefined,
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => undefined,
        "overflow-action" => "BLOCK",
        "queue-length" => "50",
        "subhandlers" => undefined
    }
}
[standalone@localhost:9999 /]
```

Add a new async log handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:add(queue-length="LENGTH")
```

Replace **HANDLER** with the name of the log handler. Replace **LENGTH** with value of the maximum number of log requests that can be held in queue.

Example 11.51.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:add(queue-length="10")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Change the log level

Use the **change-log-level** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:change-log-level(level="LEVEL")
```

Replace **HANDLER** with the name of the log handler. Replace **LEVEL** with the log level that is to be set.

Example 11.52.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:change-log-level(level="INFO")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Set the queue length

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="queue-length", value="LENGTH")
```

Replace **HANDLER** with the name of the log handler. Replace **LENGTH** with value of the maximum number of log requests that can be held in queue.

JBoss Enterprise Application Platform 6 must be restarted for this change to take effect.

Example 11.53.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:write-attribute(name="queue-length", value="150")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /]
```

Set the overflow action

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="overflow-action", value="ACTION")
```

Replace **HANDLER** with the name of the log handler. Replace **ACTION** with either **DISCARD** or **BLOCK**.

Example 11.54.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:write-attribute(name="overflow-action", value="DISCARD")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Add sub-handlers

Use the **assign-subhandler** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:assign-
subhandler(name="SUBHANDLER")
```

Replace **HANDLER** with the name of the log handler. Replace **SUBHANDLER** with the name of the log handler that is to be added as a sub-handler of this async handler.

Example 11.55.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:assign-subhandler(name="NFS_FILE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove sub-handlers

Use the **unassign-subhandler** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:unassign-
subhandler(name="SUBHANDLER")
```

Replace **HANDLER** with the name of the log handler. Replace **SUBHANDLER** with the name of the sub-handler to remove.

Example 11.56.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:unassign-subhandler(name="NFS_FILE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Remove an async log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:remove
```

Replace **HANDLER** with the name of the log handler.

Example 11.57.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

11.4. Logging Configuration Properties

11.4.1. Root Logger Properties

Table 11.5. Root Logger Properties

Property	Datatype	Description
level	string	The maximum level of log message that the root logger records.
handlers	list of strings	A list of log handlers that are used by the root logger.

[Report a bug](#)

11.4.2. Log Category Properties

Table 11.6.

Property	Datatype	Description
level	string	The maximum level of log message that the log category records.
handlers	list of strings	A list of log handlers that are used by the root logger.
use-parent-handlers	boolean	If set to true, this category will use the log handlers of the root logger in addition to any other assigned handlers.
category	string	The log category from which log messages will be captured.

[Report a bug](#)

11.4.3. Console Log Handler Properties

Table 11.7. Console Log Handler Properties

Property	Datatype	Description
level	string	The maximum level of log message the log handler records.
encoding	string	The character encoding scheme to be used for the output.
formatter	string	The log formatter used by this log handler.
target	string	The system output stream where the output of the log handler goes. This can be System.err or System.out for the system error stream or standard out stream respectively.
autoflush	boolean	If set to true the log messages will be sent to the handlers target immediately upon receipt.
name	string	The unique identifier for this log handler.

[Report a bug](#)

11.4.4. File Log Handler Properties

Table 11.8. File Log Handler Properties

Property	Type	Description
level	string	The maximum level of log message the log handler records.
encoding	string	The character encoding scheme to be used for the output.
formatter	string	The log formatter used by this log handler.
append	boolean	If set to true then 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. Changes to append require a server reboot to take effect.
autoflush	boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt. Changes to autoflush require a server reboot to take effect.
name	string	The unique identifier for this log handler.
file	object	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 .
relative-to	string	This is a property of the file object and is the directory where the log file is written to. JBoss Enterprise Application Platform 6 file path variables can be specified here. The jboss.server.log.dir variable points to the log/ directory of the server.
path	string	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.

[Report a bug](#)

11.4.5. Periodic Log Handler Properties

Table 11.9. Periodic Log Handler Properties

Property	Type	Description
append	boolean	If set to true then 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. Changes to append require a server reboot to take effect.
autoflush	boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt. Changes to append require a server reboot to take effect.
encoding	string	The character encoding scheme to be used for the output.
formatter	string	The log formatter used by this log handler.
level	string	The maximum level of log message the log handler records.
name	string	The unique identifier for this log handler.
file	object	Object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
relative-to	string	This is a property of the file object and is the directory where the log file is written to. File path variables can be specified here. The jboss.server.log.dir variable points to the log/ directory of the server.
path	string	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.
suffix	string	This is a string with is both appended to filename of the rotated logs and is used to determine the frequency of rotation. The format of the suffix is a dot (.) followed by a date string which is parsable by the java.text.SimpleDateFormat class. The log is rotated on the basis of the smallest time unit defined by the suffix. For example the suffix .yyyy-MM-dd will result in daily log rotation. Refer to http://docs.oracle.com/javase/6/docs/api/index.html?java/text/SimpleDateFormat.html

[Report a bug](#)

11.4.6. Size Log Handler Properties

Table 11.10. Size Log Handler Properties

Property	Type	Description
append	boolean	If set to true then 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. Changes to append require a server reboot to take effect.
autoflush	boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt. Changes to append require a server reboot to take effect.
encoding	string	The character encoding scheme to be used for the output.
formatter	string	The log formatter used by this log handler.
level	string	The maximum level of log message the log handler records.
name	string	The unique identifier for this log handler.
file	object	Object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
relative-to	string	This is a property of the file object and is the directory where the log file is written to. File path variables can be specified here. The <code>jboss.server.log.dir</code> variable points to the <code>log/</code> directory of the server.
path	string	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.
rotate-size	integer	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. Eg. 50m for 50 megabytes.
max-backup-index	integer	The maximum number of rotated logs that are kept. When this number is reached, the oldest log is reused.

[Report a bug](#)

11.4.7. Async Log Handler Properties

Table 11.11. Async Log Handler Properties

Property	Type	Description
level	string	The maximum level of log message the log handler records.
name	string	The unique identifier for this log handler.
Queue-length	integer	Maximum number of log messages that will be held by this handler while waiting for sub-handlers to respond.
overflow-action	string	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 behaviour as an non-async log handler. DISCARD allows the logging application to continue but the log message is deleted.
subhandlers	list of strings	This is the list of log handlers to which this async handler passes its log messages.

[Report a bug](#)

11.5. Sample XML Configuration for Logging

11.5.1. Sample XML Configuration for the Root Logger

```
<subsystem xmlns="urn:jboss:domain:logging:1.1">

  <root-logger>
    <level name="INFO"/>
    <handlers>
      <handler name="CONSOLE"/>
      <handler name="FILE"/>
    </handlers>
  </root-logger>

</subsystem>
```

[Report a bug](#)

11.5.2. Sample XML Configuration for a Log Category

```
<subsystem xmlns="urn:jboss:domain:logging:1.1">

  <logger category="com.company.accounts.rec">
    <handlers>
      <handler name="accounts-rec"/>
    </handlers>
  </logger>

</subsystem>
```

[Report a bug](#)

11.5.3. Sample XML Configuration for a Console Log Handler

```
<subsystem xmlns="urn:jboss:domain:logging:1.1">

  <console-handler name="CONSOLE">
    <level name="INFO"/>
    <formatter>
      <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
    </formatter>
  </console-handler>

</subsystem>
```

[Report a bug](#)

11.5.4. Sample XML Configuration for a File Log Handler

```
<file-handler name="accounts-rec-trail" autoflush="true">
  <level name="INFO"/>
  <file relative-to="jboss.server.log.dir" path="accounts-rec-trail.log"/>
  <append value="true"/>
</file-handler>
```

[Report a bug](#)

11.5.5. Sample XML Configuration for a Periodic Log Handler

```
<periodic-rotating-file-handler name="FILE">
  <formatter>
    <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
  </formatter>
  <file relative-to="jboss.server.log.dir" path="server.log"/>
  <suffix value=".yyyy-MM-dd"/>
  <append value="true"/>
</periodic-rotating-file-handler>
```

[Report a bug](#)

11.5.6. Sample XML Configuration for a Size Log Handler

```
<size-rotating-file-handler name="accounts_debug" autoflush="false">
  <level name="DEBUG"/>
  <file relative-to="jboss.server.log.dir" path="accounts-debug.log"/>
  <rotate-size value="500k"/>
  <max-backup-index value="5"/>
  <append value="true"/>
</size-rotating-file-handler>
```

[Report a bug](#)

11.5.7. Sample XML Configuration for a Async Log Handler

```
<async-handler name="Async_NFS_handlers">
  <level name="INFO"/>
  <queue-length value="512"/>
  <overflow-action value="block"/>
  <subhandlers>
    <handler name="FILE"/>
    <handler name="accounts-record"/>
  </subhandlers>
</async-handler>
```

[Report a bug](#)

Chapter 12. JVM

12.1. About JVM

12.1.1. About JVM Settings

Configuration of Java Virtual Machine (JVM) settings varies between the managed domain and standalone server instances. In a managed domain, the JVM settings are declared in **host.xml** and **domain.xml** configuration files, and determined by the domain controller components responsible for starting and stopping server processes. In a standalone server instance, the server startup processes can pass command line settings at startup. These can be declared from the command line or via the **System Properties** screen in the Management Console.

Managed Domain

An important feature of the managed domain is the ability to define JVM settings at multiple levels. You can configure custom JVM settings at the host level, by server group, or by server instance. The more specialized child elements will override the parent configuration, allowing for the declaration of specific server configurations without requiring exclusions at the group or host level. This also allows the parent configuration to be inherited by the other levels until settings are either declared in the configuration files or passed at runtime.

Example 12.1. JVM settings in the domain configuration file

The following example shows a JVM declaration for a server group in the **domain.xml** configuration file.

```
<server-groups>
    <server-group name="main-server-group" profile="default">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
        <socket-binding-group ref="standard-sockets"/>
    </server-group>
    <server-group name="other-server-group" profile="default">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
        <socket-binding-group ref="standard-sockets"/>
    </server-group>
</server-groups>
```

In this instance a server group called **main-server-group** is declaring a heap size of 64 megabytes, and a maximum heap size of 512 megabytes. Any server that belongs to this group will inherit these settings. You can change these settings for the group as a whole, by the host, or the individual server.

Example 12.2. Domain settings in the host configuration file

The following example shows a JVM declaration for a server group in the `host.xml` configuration file.

```
<servers>
    <server name="server-one" group="main-server-group" auto-start="true">
        <jvm name="default"/>
    </server>
    <server name="server-two" group="main-server-group" auto-start="true">
        <jvm name="default">
            <heap size="64m" max-size="256m"/>
        </jvm>
        <socket-binding-group ref="standard-sockets" port-offset="150"/>
    </server>
    <server name="server-three" group="other-server-group" auto-
start="false">
        <socket-binding-group ref="standard-sockets" port-offset="250"/>
    </server>
</servers>
```

In this instance, a server named `server-two` belongs to the server group named `main-server-group`, inheriting the JVM settings from the `default` JVM group. In the previous example, the main heap size for `main-server-group` was set at 512 megabytes. By declaring the lower maximum heap size of 256 megabytes, `server-two` can override the `domain.xml` settings to fine-tune performance as desired.

Standalone server settings at runtime

The JVM settings for standalone 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 at the Linux command-line is:

```
[user@host bin]$ export JAVA_OPTS="-Xmx1024M"
```

The same setting can be used in a Microsoft Windows environment, as follows:

```
C:\> set JAVA_OPTS="Xmx1024M"
```

Alternatively, JVM settings can be added to the `standalone.conf` file found in the `EAP_HOME/bin` folder, which contains examples of options to pass to the JVM.

[Report a bug](#)

12.1.2. Display the JVM Status in the Management Console

Prerequisites

- ▶ [Section 2.6.2. "Start JBoss Enterprise Application Platform 6 as a Standalone Server"](#)
- ▶ [Section 2.6.3. "Start JBoss Enterprise Application Platform 6 as a Managed Domain"](#)
- ▶ [Section 3.2.2. "Log in to the Management Console"](#)

Java Virtual Machine (JVM) status can be displayed in the Management Console for either the standalone server or a managed domain. The console shows the heap usage, non heap usage, and thread usage of the server in units of megabytes. While the statistics are not displayed in real-time, you can refresh the console display to provide an up-to-date overview of JVM resources.

The JVM status shows the following values.

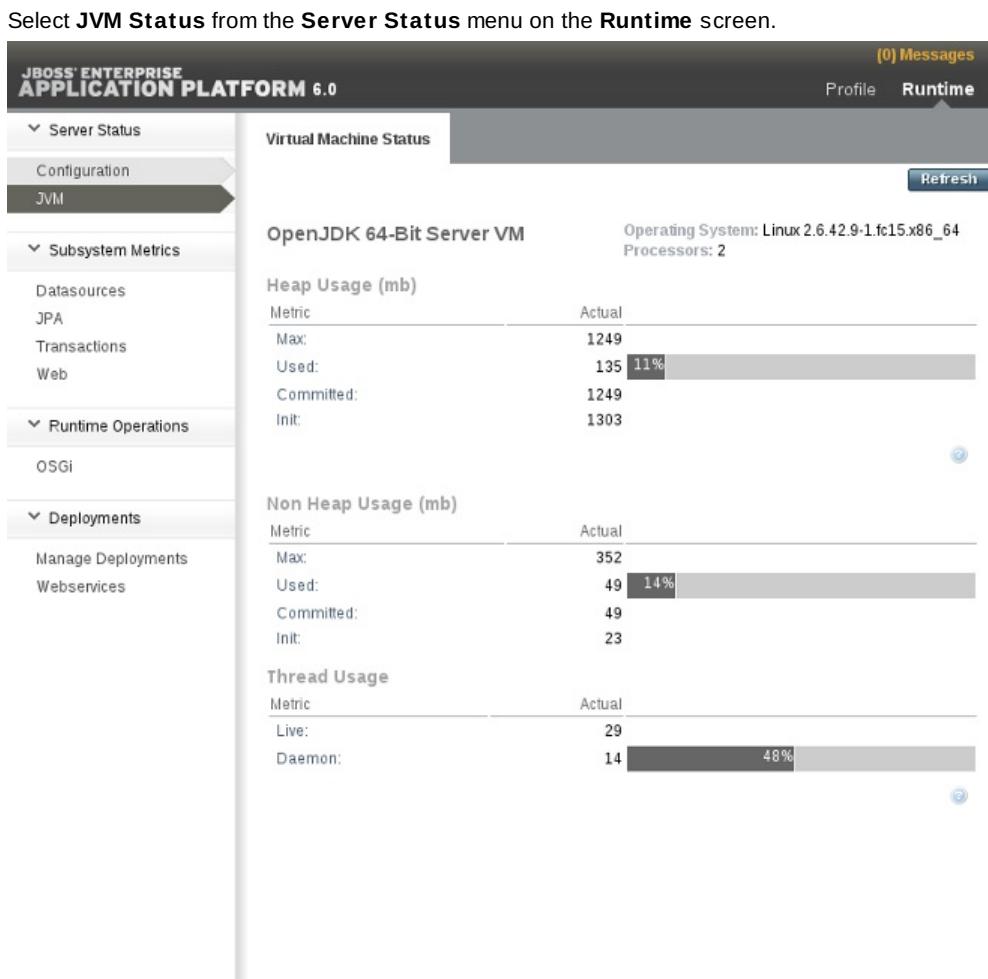
Table 12.1. JVM Status Attributes

Type	Description
Max	The maximum amount of memory in bytes that can be used for memory management.
Used	The amount of used memory in mega bytes.
Committed	The amount of memory in bytes that is committed for the Java virtual machine to use.
Init	The amount of memory in bytes that the Java virtual machine initially requests from the operating system for memory management.

» **Display the JVM status**

You can view the JVM status in either the standalone server instance or a managed domain.

A. Display the JVM status for a standalone server instance

**Figure 12.1. JVM status for a standalone server instance**

B. Display the JVM status for a managed domain

Select JVM Status from the Domain Status menu on the Runtime screen.

- C. The managed domain can provide visibility of all server instances in the server group, but will only allow you to view one server at a time by selecting from the server menu. To view the status of other servers in your server group, click on the drop-down box at the top left of the screen to select from the host and servers displayed in your group and click the **Done** button to load the results.

Result

The status of the JVM settings for the server instance are displayed.

[Report a bug](#)

Chapter 13. HTTP Clustering and Load Balancing

13.1. Introduction

13.1.1. About High-Availability and Load Balancing Clusters

Clustering refers to using multiple resources, such as servers, as though they were a single entity. The two main types of clustering are *Load balancing (LB)* and *High-availability (HA)*. In a LB cluster, all resources run at the same time, and a management layer spreads the work load across them.

In HA clustering, one resource runs, and another is available to step in if the first one becomes unavailable. The purpose of HA clustering is to reduce the consequences of hardware, software, or network outages.

The JBoss Enterprise Application Platform supports clustering at several different levels. Some of the subsystems that can be made highly available are:

- ▶ Instances of the Application Server
- ▶ The web server, whether it is the internal JBoss Web server, Apache HTTPD, Microsoft IIS, Oracle iPlanet Web Server, or Apache Tomcat
- ▶ Stateful, stateless, and entity Enterprise JavaBeans (EJBs)
- ▶ JNDI services
- ▶ Single Sign On (SSO) Mechanisms
- ▶ Distributed cache
- ▶ HTTP sessions
- ▶ JMS services and Message-driven beans (MDBs)

[Report a bug](#)

13.1.2. Components Which Can Benefit from High Availability

High Availability (HA) falls into a few broad categories in the JBoss Enterprise Application Platform.

The Container

Several instances of the Enterprise Application Server (running as a standalone server) or a server group's members (running as part of a managed domain) can be configured to be highly available. This means that if one instance or member is stopped or disappears from the cluster, its work load is moved to a peer. The work load can be managed in such a way to provide load-balancing functionality as well, so that servers or server groups with more or better resources can take on a larger portion of the work load, or additional capacity can be added during times of high load.

The Web Server

The web server itself can be clustered for HA, using one of several compatible load balancing mechanisms. The most flexible is `mod_cluster` connector, which is tightly integrated with the JBoss Enterprise Application Platform's container. Other choices include Apache `mod_jk` or `mod_proxy` connectors, or the ISAPI and NSAPI connectors.

The Application

Deployed applications can be made highly-available because of the Java Enterprise Edition 6 (Java EE 6) specification. Stateless or stateful session EJBs can be clustered so that if the node which is involved in the work disappears, another node can take over, and in the case of stateful session beans, preserve the state.

[Report a bug](#)

13.1.3. Overview of HTTP Connectors

JBoss Enterprise Application Platform has the ability to use load-balancing and high-availability mechanisms built into external HTTPD web servers, such as Apache Web Server, Microsoft IIS, and Oracle iPlanet. The JBoss Enterprise Application Platform communicates with the external web server using a HTTP Connector. These HTTP connectors are configured within the web subsystem of JBoss Enterprise Application Platform.

The HTTPD servers include software modules which control the way that HTTP requests are routed to

JBoss Enterprise Application Platform worker 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 Enterprise Application Platform server nodes, to move work loads to alternate servers in case of a failure event, or both. These two abilities are called *Load Balancing* and *High Availability (HA)*.

The JBoss Enterprise Application Platform supports several different HTTP connectors. The one you choose depends on the HTTPD you connect to and the other functionality you need.

The table below lists the differences between the different HTTP connectors which are compatible with the JBoss Enterprise Application Platform. For the most up-to-date information about supported configurations for HTTP connectors, refer to <https://access.redhat.com/kb/docs/DOC-34454>.

Table 13.1. HTTP connector features and constraints

Connector	Web server	Supported operating systems	Supported protocols	Adapts to deployment status	Supports sticky session
mod_cluster	JBoss Enterprise Web Server HTTPD	Red Hat Enterprise Linux, Microsoft Windows Server, Solaris	HTTP, HTTPS, AJP	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.	Yes
mod_jk	JBoss Enterprise Web Server HTTPD, Apache HTTPD	Red Hat Enterprise Linux, Microsoft Windows Server if using JBoss Enterprise Web Server, Solaris	AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes
mod_proxy	JBoss Enterprise Web Server HTTPD, Apache HTTPD	Red Hat Enterprise Linux, Microsoft Windows Server if using JBoss Enterprise Web Server, Solaris	HTTP, HTTPS	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes
ISAPI	Microsoft IIS	Microsoft Windows Server	AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes
NSAPI	Sun Java System Web Server	Oracle Solaris	AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes

Learn more about each HTTP Connector

- ▶ [Section 13.3.1, “About the mod_cluster HTTP Connector”](#)
- ▶ [Section 13.4.1, “About the Apache mod_jk HTTP Connector”](#)
- ▶ [Section 13.5.1, “About the Apache mod_proxy HTTP Connector”](#)
- ▶ [Section 13.6.1, “About the Internet Server API \(ISAPI\) HTTP Connector”](#)
- ▶ [Section 13.7.1, “About the Netscape Server API \(NSAPI\) HTTP Connector”](#)

[Report a bug](#)

13.1.4. Worker Node

HTTP connector node

A *worker node*, sometimes referred to as simply a *node*, is a JBoss Enterprise Application Platform server which accepts requests from one or more client-facing HTTPD servers. The JBoss Enterprise

Application Platform can accept requests from its own HTTPD, the HTTPD shipped with JBoss Enterprise Web Server, Apache HTTPD, Microsoft IIS, or Oracle iPlanet Web Server (Formerly Netscape Web Server).

For an overview of HTTP connectors supported by the JBoss Enterprise Application Platform and how to configure them, refer to [Section 13.1.3, “Overview of HTTP Connectors”](#).

Cluster node

A cluster node is a member of a cluster of servers. Such a cluster may be load-balancing, high-availability, or both. In a load-balancing cluster, a central manager distributes work loads amongst its nodes equally, by some situation-specific measurement of equality. In a high-availability cluster, some nodes are actively doing work, and others are waiting to step in if one of the active nodes leaves the cluster.

[Report a bug](#)

13.2. General Configuration

13.2.1. Subsystem Configuration Overview

Introduction

JBoss Enterprise Application Platform 6 uses a simplified configuration, with one configuration file per domain or per standalone server. In a standalone domain, a separate file exists for each host controller as well. Changes to the configuration persist automatically, so XML should not be edited by hand. The configuration is scanned and overwritten automatically by the Management API. The command-line based Management CLI and web-based Management Console allow you to configure each aspect of JBoss Enterprise Application Platform.

JBoss Enterprise Application Platform 6 is built on the concept of modular classloading. Each API or service provided by the Platform is implemented as a module, which is loaded and unloaded on demand. Most modules include a configurable element called a subsystem. Subsystem configuration information is stored in the unified configuration file `EAP_HOME/domain/configuration/domain.xml` for a managed domain or `EAP_HOME/standalone/configuration/standalone.xml` for a standalone server. Many of the subsystems include configuration details that were configured via deployment descriptors in previous versions of JBoss Enterprise Application Platform.

Subsystem Configuration Schemas

Each subsystem's configuration is defined in an XML schema. The configuration schema are located in the `EAP_HOME/docs/schema/` directory of your installation.

The following subsystems are known as *simple subsystems*, because they do not have any configurable attributes or elements. They are generally listed at the top of the configuration file.

Simple Subsystems

- ▶ `ee`– the Java EE 6 API implementation
- ▶ `ejb`– the Enterprise JavaBeans (EJB) subsystem
- ▶ `jaxrs`– the JAX-RS API, provided by RESTeasy.
- ▶ `sar`– the subsystem which supports Service Archives.
- ▶ `threads`– the subsystem which supports process threads.
- ▶ `weld`– the Contexts and Dependency Injection API, provided by Weld.

[Report a bug](#)

13.2.2. Configure the Web Subsystem

You can configure most aspects of the Web subsystem using the web-based Management Console or the command-line Management CLI. Each setting is explained in the order it appears in the Management Console, and Management CLI commands are also provided.

View the Web Subsystem Using the Management Console

To configure the Web Subsystem using the web-based Management Console, click the **Profiles** tab at the top right. For a managed domain, select the server profile you wish to configure from the **Profile**

selection box at the top left. Expand the **Subsystems** menu, then expand the **Web** menu. Each configurable part of the Web subsystem is shown.



Note

The **mod_cluster** component is only available if your profile is **ha** or **full-ha**, in a managed domain, or if you start your standalone server with the **standalone-ha** profile. **mod_cluster** configuration is covered in [Section 13.3.2, “Configure the mod_cluster Subsystem”](#).

Configure the JSP Container, HTTP Connectors, and Virtual HTTP Servers

To configure the JSP Container, HTTP connectors, and virtual HTTP servers, click the **Servlet/HTTP** menu entry. Click the **Edit** button to change any values. Click the **Advanced** button to view advanced options. The options are explained below. Options for HTTP connectors and virtual servers are shown in separate tables.

Table 13.2. Servlet/HTTP Configuration Options

Option	Description	CLI Command
Disabled?	If true , disables the Java ServerPages (JSP) container. Defaults to false . This is useful if you do not use any Java ServerPages (JSPs).	<code>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=disabled,value=false)</code>
Development?	If true, enables Development Mode, which produces more verbose debugging information. Defaults to false .	<code>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=development,value=false)</code>
Keep Generated?	Click Advanced to see this option, if it is hidden. If true keeps generated Servlets. Enabled by default.	<code>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=keep-generated,value=true)</code>
Check Interval?	Click Advanced to see this option, if it is hidden. A value in seconds, which determines how often to check for JSP updates using a background process. Defaults to 0 .	<code>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=check-interval,value=0)</code>
Display Source?	Click Advanced to see this option, if it is hidden. If true , the JSP source fragment is displayed when a runtime error occurs. Defaults to true .	<code>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=display-source-fragment,value=true)</code>

HTTP connectors are used for load balancing and High Availability clusters such as **mod_cluster**, **mod_jk**, **mod_proxy**, **ISAPI**, and **NSAPI**. To configure a HTTP connector, select the **Connectors** tab and click **Add**. To remove a HTTP connector, select its entry and click **Remove**. To edit a HTTP connector, select its entry and click **Edit**.

When you create a new connector using the Management CLI, its options are all set at once, as in the

following command:

Example 13.1. Create a New Connector

```
/profile=full-ha/subsystem=web/connector=ajp/:add(socket-
binding=ajp,scheme=http,protocol=AJP/1.3,secure=false,name=ajp,max-post-
size=2097152,enabled=true,enable-lookups=false,redirect-port=8433,max-save-post-
size=4096)
```

Table 13.3. Connector Options

Option	Description	CLI Command
Name	A unique name for the connector, for display purposes.	<pre>/profile=full- ha/subsystem=web/connec- tor=ajp/:write- attribute(name=name, val- ue=ajp)</pre>
Socket Binding	The named socket binding the connector should bind to. A socket binding is a mapping between a socket name and a network port. Socket bindings are configured for each standalone server, or via socket binding groups in a managed domain. A socket binding group is applied to a server group.	<pre>/profile=full- ha/subsystem=web/connec- tor=ajp/:write- attribute(name=socket- binding,value=ajp)</pre>
Scheme	The web connector scheme, such as HTTP or HTTPS.	<pre>/profile=full- ha/subsystem=web/connec- tor=ajp/:write- attribute(name=scheme, v- alue=http)</pre>
Protocol	The web connector protocol to use, such as AJP or HTTP.	<pre>/profile=full- ha/subsystem=web/connec- tor=ajp/:write- attribute(name=protocol ,value=AJP/1.3)</pre>
Enabled	Whether or not this web connector is enabled.	<pre>/profile=full- ha/subsystem=web/connec- tor=ajp/:write- attribute(name=enabled, value=true)</pre>

To configure virtual servers, click the **Virtual Servers** tab. Use the **Add** button to add a new virtual server. To edit or remove a virtual server, select its entry and click the **Edit** or **Remove** button.

When you add a new virtual server using the Management CLI, all required options are set at once, as in the following command.

Example 13.2. Add a New Virtual Server

```
/profile=full-ha/subsystem=web/virtual-server=default-host/:add(enable-welcome-
root=true,default-web-
module=ROOT.war,alias=["localhost","example.com"],name=default-host)
```

Table 13.4. Virtual Servers Options

Option	Description	CLI Command
Name	A unique name for the virtual server, for display purposes.	<pre>/profile=full- ha/subsystem=web/virtua l-server=default- host/:write- attribute(name=name, val ue=default-host)</pre>
Alias	A list of hostnames which should match this virtual server. In the Management Console, use one hostname per line.	<pre>/profile=full- ha/subsystem=web/virtua l-server=default- host/:write- attribute(name=alias, val ue=["localhost", "exampl e.com"])</pre>
Default Module	The module whose web application should be deployed at the root node of this virtual server, and will be displayed when no directory is given in the HTTP request.	<pre>/profile=full- ha/subsystem=web/virtua l-server=default- host/:write- attribute(name=default- web- module, value=ROOT.war)</pre>

Configure Web Services Options

To configure Web Services options, click the **Web Services** menu item. The options are explained in the table below.

Table 13.5. Web Services Configuration Options

Option	Description	CLI Command
Modify WSDL Address	Whether the WSDL address can be modified by applications. Defaults to true .	<pre>/profile=full- ha/subsystem=webservices/:write- attribute(name=modify- wsdl- address,value=true)</pre>
WSDL Host	The WSDL contract of a JAX-WS Web Service includes a <soap:address> element which points to the location of the endpoint. If the value of <soap:address> is a valid URL, it is not overwritten unless modify-wsdl-address is set to true . If the value of <soap:address> is not a valid URL, it is overwritten using the values of wsdl-host and either wsdl-port or wsdl-secure- port . If wsdl-host is set to jbossws.undefined.host , the requester's host address is used when the <soap-address> is rewritten. Defaults to \${jboss.bind.address:12 7.0.0.1} , which uses 127.0.0.1 if no bind address is specified when JBoss Enterprise Application Platform is started.	<pre>/profile=full- ha/subsystem=webservices/:write- attribute(name=wsdl- host,value=\${jboss.bind .address:12.0.0.1})</pre>
WSDL Port	The non-secure port that is used to rewrite the SOAP address. If this is set to 0 (the default), the port is identified by querying the list of installed connectors.	<pre>/profile=full- ha/subsystem=webservices/:write- attribute(name=wsdl- port,value=80)</pre>
WSDL Secure Port	The secure port that is used to rewrite the SOAP address. If this is set to 0 (the default), the port is identified by querying the list of installed connectors.	<pre>/profile=full- ha/subsystem=webservices/:write- attribute(name=wsdl- secure-port,value=443)</pre>

[Report a bug](#)

13.2.3. Implement SSL Encryption for the JBoss Enterprise Application Platform Web Server

Introduction

Many web applications require a SSL-encrypted connection between clients and server, also known as a **HTTPS** connection. You can use this procedure to enable **HTTPS** on your server or server group.

Prerequisites

- ▶ You need a set of SSL encryption keys and a SSL encryption certificate. You may purchase these from a certificate-signing authority, or you can generate them yourself using command-line utilities. To generate encryption keys using Red Hat Enterprise Linux utilities, refer to [Section 13.2.4, “Generate a SSL Encryption Key and Certificate”](#).

- ▶ You need to know the following details about your specific environment and set-up:
 - The full directory name and path to your certificate files
 - The encryption password for your encryption keys.
- ▶ You need to run the Management CLI and connect it to your domain controller or standalone server.



Note

This procedure uses commands appropriate for a JBoss Enterprise Application Platform configuration that uses a managed domain. If you use a standalone server, modify Management CLI commands by removing the **/profile=default** from the beginning of any Management CLI commands.

Procedure 13.1. Configure the JBoss Web Server to use HTTPS

1. Add a new HTTPS connector.

Execute the following Management CLI command, changing the profile as appropriate. This creates a new secure connector, called **HTTPS**, which uses the **https** scheme, the **https** socket binding (which defaults to **8443**), and is set to be secure.

Example 13.3. Management CLI Command

```
/profile=default/subsystem=web/connector=https/:add(socket-
binding=https, scheme=https, protocol=HTTP/1.1, secure=true)
```

2. Configure the SSL encryption certificate and keys.

Execute the following CLI commands to configure your SSL certificate, substituting your own values for the example ones. This example assumes that the keystore is copied to the server configuration directory, which is **EAP_HOME/domain/configuration/** for a managed domain.

Example 13.4. Management CLI Command

```
/profile=default/subsystem=web/connector=https/ssl=configuration:add(name=ht
tps,certificate-key-
file=${jboss.server.config.dir}/keystore.jks,password=SECRET, key-
alias=KEY_ALIAS)
```

For a full listing of parameters you can set for the SSL properties of the connector, refer to [Section 13.2.5, “SSL Connector Reference”](#).

3. Deploy an application.

Deploy an application to a server group which uses the profile you have configured. If you use a standalone server, deploy an application to your server. HTTP requests to it use the new SSL-encrypted connection.

[Report a bug](#)

13.2.4. Generate a SSL Encryption Key and Certificate

To use a SSL-encrypted HTTP connection (HTTPS), as well as other types of SSL-encrypted communication, you need a signed encryption certificate. You can purchase a certificate from a Certificate Authority (CA), or you can use a self-signed certificate. Self-signed certificates are not considered trustworthy by many third parties, but are appropriate for internal testing purposes.

This procedure enables you to create a self-signed certificate using utilities which are available on Red Hat Enterprise Linux.

Prerequisites

- ▶ You need the **keytool** utility, which is provided by any Java Development Kit implementation. OpenJDK on Red Hat Enterprise Linux installs this command to **/usr/bin/keytool**.
- ▶ Understand the syntax and parameters of the **keytool** command. This procedure uses extremely generic instructions, because further discussion of the specifics of SSL certificates or the **keytool** command are out of scope for this documentation.

Procedure 13.2. Task

1. Generate a keystore with public and private keys.

Run the following command to generate a keystore named **server.keystore** with the alias **jboss** in your current directory.

```
keytool -genkey -alias jboss -keyalg RSA -keystore server.keystore -storepass
mykeystorepass --dname
"CN=jsmith, OU=Engineering, O=mycompany.com, L=Raleigh, S=NC, C=US"
```

The following table describes the parameters used in the keytool command:

Parameter	Description
-genkey	The keytool command to generate a key pair containing a public and private key.
-alias	The alias for the keystore. This value is arbitrary, but the alias jboss is the default used by the JBoss Web server.
-keyalg	The key pair generation algorithm. In this case it is RSA .
-keystore	The name and location of the keystore file. The default location is the current directory. The name you choose is arbitrary. In this case, the file will be named server.keystore .
-storepass	This password is used to authenticate to the keystore so that the key can be read. The password must be at least 6 characters long and must be provided when the keystore is accessed. In this case, we used mykeystorepass . If you omit this parameter, you will be prompted to enter it when you execute the command.
-keypass	This is the password for the actual key.
<div style="background-color: #6B8E23; color: white; padding: 5px; text-align: center;">  Note </div> <p>Due to an implementation limitation this must be the same as the store password.</p>	
--dname	A quoted string describing the distinguished name for the key, for example: "CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,C=US". This string is a concatenation of the following components: <ul style="list-style-type: none"> ▶ CN - The common name or host name. If the hostname is "jsmith.mycompany.com", the CN is "jsmith". ▶ OU - The organizational unit, for example "Engineering" ▶ O - The organization name, for example "mycompany.com". ▶ L - The locality, for example "Raleigh" or "London" ▶ S - The state or province, for example "NC". This parameter is optional. ▶ C - The 2 letter country code, for example "US" or "UK",

When you execute the above command, you are prompted for the following information:

- ▶ If you did not use the **-storepass** parameter on the command line, you are asked to enter the keystore password. Re-enter the new password at the next prompt.
- ▶ If you did not use the **-keypass** parameter on the command line, you are asked to enter the key password. Press **Enter** to set this to the same value as the keystore password.

When the command completes, the file **server.keystore** now contains the single key with the alias **jboss**.

2. Verify the key.

Verify that the key works properly by using the following command.

```
keytool -list -keystore server.keystore
```

You are prompted for the keystore password. The contents of the keystore are displayed (in this case, a single key called **jboss**). Notice the type of the **jboss** key, which is **keyEntry**. This indicates that the keystore contains both a public and private entry for this key.

3. Generate a certificate signing request.

Run the following command to generate a certificate signing request using the public key from the keystore you created in step 1.

```
keytool -certreq -keyalg RSA -alias jboss -keystore server.keystore -file certreq.csr
```

You are prompted for the password in order to authenticate to the keystore. The **keytool** command then creates a new certificate signing request called **certreq.csr** in the current working directory.

4. Test the newly generated certificate.

Test the contents of the certificate by using the following command.

```
openssl req -in certreq.csr -noout -text
```

The certificate details are shown.

5. Optional: Submit your certificate to a Certificate Authority (CA).

A Certificate Authority (CA) can authenticate your certificate so that it is considered trustworthy by third-party clients. The CA supplies you with a signed certificate, and optionally with one or more intermediate certificates.

6. Optional: Export a self-signed certificate from the keystore.

If you only need it for testing or internal purposes, you can use a self-signed certificate. You can export one from the keystore you created in step 1 as follows:

```
keytool -export -alias jboss -keystore server.keystore -file server.crt
```

You are prompted for the password in order to authenticate to the keystore. A self-signed certificate, named **server.crt**, is created in the current working directory.

7. Import the signed certificate, along with any intermediate certificates.

Import each certificate, in the order that you are instructed by the CA. For each certificate to import, replace **intermediate.ca** or **server.crt** with the actual file name. If your certificates are not provided as separate files, create a separate file for each certificate, and paste its contents into the file.

Note

Your signed certificate and certificate keys are valuable assets. Be cautious with how you transport them between servers.

```
keytool -import -keystore server.keystore -alias intermediateCA -file intermediate.ca
```

```
keytool -import -alias jboss -keystore server.keystore -file server.crt
```

8. Test that your certificates imported successfully.

Run the following command, and enter the keystore password when prompted. The contents of your keystore are displayed, and the certificates are part of the list.

```
keytool -list -keystore server.keystore
```

Result

Your signed certificate is now included in your keystore and is ready to be used to encrypt SSL connections, including HTTPS web server communications.

[Report a bug](#)

13.2.5. SSL Connector Reference

JBoss Web connectors may include the following SSL configuration attributes. The CLI commands provided are designed for a managed domain using profile **default**. Change the profile name to the one you wish to configure, for a managed domain, or omit the **/profile=default** portion of the command, for a standalone server.

Table 13.6. SSL Connector Attributes

Attribute	Description	CLI Command
Name	The display name of the SSL connector.	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=name, value=https))</pre>
verify-client	Set to true to require a valid certificate chain from the client before accepting a connection. Set to want if you want the SSL stack to request a client Certificate, but not fail if one is not presented. Set to false (the default) to not require a certificate chain unless the client requests a resource protected by a security constraint that uses CLIENT -CERT authentication.	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=verify-client, value=want))</pre>
verify-depth	The maximum number of intermediate certificate issuers checked before deciding that the clients do not have a valid certificate. The default value is 10 .	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=verify-depth, value=10))</pre>
certificate-key-file	The full file path and file name of the keystore file where the signed server certificate is stored. With JSSE encryption, this certificate file will be the only one, while OpenSSL uses several files. The default value is the .keystore file in the home directory of the user running JBoss Enterprise Application Platform. If your keystoreType does not use a file, set the parameter to an empty string.	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=certificate-key-file, value=../domain/configuration/server.keystore))</pre>
certificate-file	If you use OpenSSL encryption, set the value of this parameter to the path to the file containing the server certificate.	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=certificate-file, value=server.crt))</pre>
password	The password for both the trustore and keystore. The default value is changeit , so you must change it to match the password of your keystore for your configuration to work.	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=password, value=changeit))</pre>
protocol	The version of the SSL protocol	

protocol	The version of the SSL protocol to use. Supported values include SSLv2 , SSLv3 , TLSv1 , SSLv2+SSLv3 , and ALL . The default is ALL .	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=protocol,value=ALL))
cipher-suite	A comma-separated list of the encryption ciphers which are allowed. The JVM default for JSSE contains weak ciphers which should not be used. The example only lists two possible ciphers, but real-world examples will likely use more.	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=cipher-suite,value="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA"))
key-alias	The alias used to for the server certificate in the keystore. The default value is jboss .	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=key-alias,value=jboss))
truststore-type	The type of the truststore. Various types of keystores are available, including PKCS12 and Java's standard JKS .	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=truststore-type,value=jks))
keystore-type	The type of the keystore. Various types of keystores are available, including PKCS12 and Java's standard JKS .	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=keystore-type,value=jks))
ca-certificate-file	The file containing the CA certificates. This is the truststoreFile , in the case of JSSE, and uses the same password as the keystore. The ca-certificate-file file is used to validate client certificates.	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=certificate-file,value=ca.crt))
ca-certificate-password	The Certificate password for the ca-certificate-file . In the example below, replace the password with your own masked password.	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=ca-certificate-password,value=MASKED_PASSWORD))
ca-revocation-url	A file or URL which contains the revocation list. It refers to the crlFile for JSSE or the SSLCARevocationFile for SSL.	/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(attribute(name=ca-revocation-url,value=ca.crl))
session-cache-size	The size of the SSL Session	

session-cache-size	The size of the session cache. The default is 0 , which disables the session cache.	/profile=default/subsys tem=web/connector=HTTPSS /ssl=configuration/:wri te- attribute(name=session-cache-size, value=100)
session-timeout	The number of seconds before a cached SSLSession expires. The default is 86400 seconds, which is 24 hours.	/profile=default/subsys tem=web/connector=HTTPSS /ssl=configuration/:wri te- attribute(name=truststore-type, value=43200)

[Report a bug](#)

13.2.6. About Web Service Endpoints

Web service endpoints are exposed in JBoss Enterprise Application Platform 6 through the deployments that provide the endpoint implementation. This allows the endpoints to be queried as deployment resources. Each endpoint requires a web context and a WSDL URL to facilitate web service requests, indicating a specific location for associating the binding attributes and an instance of a service.

The web service subsystem is provided by the JBossWS project. For a detailed description of the available configuration properties, please consult the project documentation.

- ▶ JBossWS homepage: <http://www.jboss.org/jbossws>
- ▶ Project Documentation: <https://docs.jboss.org/author/display/JBWS>

[Report a bug](#)

13.2.7. Replace the Default Welcome Web Application

JBoss Enterprise Application Platform 6 includes a Welcome application, which displays when you open the URL of the server at port 8080. You can replace this application with your own web application by following this procedure.

Procedure 13.3. Task

1. Disable the Welcome application.

Use the Management CLI script **EAP_HOME/bin/jboss-cli.sh** to run the following command. You may need to change the profile to modify a different managed domain profile, or remove the **/profile=default** portion of the command for a standalone server.

```
/profile=default/subsystem=web/virtual-server=default-host:write-attribute(name=enable-welcome-root, value=false)
```

2. Configure your Web application to use the root context.

To configure your web application to use the root context (/) as its URL address, modify its **jboss-web.xml**, which is located in the **META-INF/** or **WEB-INF/** directory. Replace its **<context-root>** directive with one that looks like the following.

```
<jboss-web>
  <context-root>/</context-root>
</jboss-web>
```

3. Deploy your application.

Deploy your application to the server group or server you modified in the first step. The application is now available on **http://SERVER_URL:PORT/**.

[Report a bug](#)

13.2.8. About the Stand-Alone HTTPD

JBoss Enterprise Application Platform is tested and supported with the Apache HTTPD which is

included with certified versions of Red Hat Enterprise Linux 6. Apache HTTPD is also available for other configurations, such as Microsoft Windows Server. However, since Apache HTTPD is a separate product produced by the Apache Foundation, it used to be difficult to be sure that the version of Apache HTTPD a customer used was compatible with JBoss Enterprise Application Platform.

A stand-alone Apache HTTPD bundle is now included as a separate download with JBoss Enterprise Application Platform 6. This simplifies installation and configuration in environments other than Red Hat Enterprise Linux, or on systems which already have a configured HTTPD and want to use a separate instance for web applications. You can download this HTTPD as a separate download in the Customer Service Portal, listed under the available JBoss Enterprise Application Platform 6 downloads for your installation platform.

[Report a bug](#)

13.2.9. Install the Apache HTTPD included with JBoss Enterprise Application Platform 6

Prerequisites

- » You need root or administrator access to complete this task.

Procedure 13.4. Task

1. **Navigate to the JBoss Enterprise Application Platform downloads list for your platform, on the Red Hat Customer Service Portal.**

Log in to the Red Hat Customer Service Portal at <https://access.redhat.com>. Using the menus at the top, choose **Downloads, JBoss Enterprise Middleware Downloads**. Select **Application Platform** from the combo box. Another combo box appears. Select the version of the JBoss Enterprise Application Platform to see available downloads for that version.

2. **Choose the HTTPD binary from the list.**

Find the HTTPD binary for your operating system and architecture. Click the **Download** link. A ZIP file containing the HTTPD distribution downloads to your computer.

3. **Extract the ZIP to the system where the HTTPD binary will run.**

Extract the ZIP to your preferred web server, to a location of your choice. It often makes sense to place this inside the directory where you installed the JBoss Enterprise Application Platform, commonly referred to as **EAP_HOME**. In this case, your HTTPD would be located in **EAP_HOME/httpd/**. You can now use this location for **HTTPD_HOME**, as found in other JBoss Enterprise Application Platform documentation.

4. **Configure the HTTPD.**

Configure the HTTPD to meet the needs of your organization. You can use the documentation available from the Apache Foundation at <http://httpd.apache.org/> for general guidance.

5. **Start the HTTPD.**

Start the HTTPD using the following command:

```
EAP_HOME/sbin/apachectl start
```

6. **Stop the HTTPD.**

To stop the HTTPD, issue the following command:

```
EAP_HOME/sbin/apachectl stop
```

[Report a bug](#)

13.2.10. Use an External HTTPD as the Web Front-end for JBoss Enterprise Application Platform Applications

Overview

For reasons to use an external HTTPD as the web front-end, as well as advantages and disadvantages of the different HTTP connectors supported by the JBoss Enterprise Application Platform, refer to [Section 13.1.3, “Overview of HTTP Connectors”](#). In some situations, you can use the HTTPD that comes with your operating system. Otherwise, you can use the HTTPD that ships as part of JBoss Enterprise Web Server.

After you have decided which HTTPD and HTTP connector to use, refer to one of the following procedures:

- ▶ [Section 13.2.9, “Install the Apache HTTPD included with JBoss Enterprise Application Platform 6”](#)
- ▶ [Section 13.3.3, “Install the mod_cluster Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#)
- ▶ [Section 13.4.3, “Install the Mod_jk Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#)
- ▶ [Section 13.6.2, “Configure Microsoft IIS to Use the ISAPI Redirector”](#)
- ▶ [Section 13.7.2, “Configure the NSAPI Connector on Oracle Solaris”](#)
- ▶ [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#)

[Report a bug](#)

13.2.11. Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD

Overview

The JBoss Enterprise Application Platform does not need to know which proxy it is accepting requests from, only the port and protocol to look for. This is not true of **mod_cluster**, which is more tightly coupled to the configuration of the JBoss Enterprise Application Platform. But the following task works for **mod_jk**, **mod_proxy**, **ISAPI**, and **NSAPI**. Substitute the protocols and ports you need to configure with the ones in the examples.

To configure the JBoss Enterprise Application Platform for **mod_cluster**, refer to [Section 13.3.5, “Configure a mod_cluster Worker Node”](#).

Prerequisites

- ▶ [Section 13.5.2, “Install the Mod_proxy HTTP Connector Into Apache HTTPD”](#)
- ▶ You need to be logged into the Management CLI or Management Console to perform this task. The exact steps in the task use the Management CLI, but the same basic procedure is used in the Management Console.
- ▶ You need a list of which protocols you will be using, whether HTTP, HTTPS, or AJP.

Procedure 13.5. Task

1. Configure the **jvmRoute** and **useJK** system properties.

Use the following two commands to configure the **jvmRoute** and **useJK** system properties, which are necessary for HTTP connectors to work properly. Replace **JVM_ROUTE** with your own node name.

```
[user@localhost:9999 /] /system-property=jvmRoute/:add(value=JVM_ROUTE,boot-time=true)
```

```
[user@localhost:9999 /] /system-property=UseJK/:add(value=true,boot-time=true)
```

2. List the connectors available in the web subsystem.

Note

This step is only necessary if you are not using the **standalone.ha.xml** configuration for a standalone server, or the **ha** or **full-ha** profiles for a server group in a managed domain. Those configurations already include all of the necessary connectors.

In order for an external HTTPD to be able to connect to the JBoss Enterprise Application Platform's web server, the web subsystem needs a connector. Each protocol needs its own connector, which is tied to a socket group.

To list the connectors currently available, issue the following command:

```
[standalone@localhost:9999 /] /subsystem=web:read-children-names(child-type=connector)
```

If there is no line indicating the connector you need (HTTP, HTTPS, AJP), you need to add the connector.

3. Read the configuration of a connector.

To see the details of how a connector is configured, you can read its configuration. The following command reads the configuration of the AJP connector. The other connectors have similar configuration output.

```
[standalone@localhost:9999 /] /subsystem=web/connector=ajp:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "enable-lookups" => false,
        "enabled" => true,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "protocol" => "AJP/1.3",
        "redirect-port" => 8443,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "ajp",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}
```

4. Add the necessary connectors to the web subsystem.

To add a connector to the web subsystem, it needs to have a socket binding. The socket binding is added to the socket binding group used by your server or server group. The following steps assume that your server group is **server-group-one** and that your socket binding group is **standard-sockets**.

a. Add a socket to the socket binding group.

To add a socket to the socket binding group, issue the following command, replacing the protocol and port with the ones you need.

```
[standalone@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=ajp:add(port=8009)
```

b. Add the socket binding to the web subsystem.

Issue the following command to add a connector to the web subsystem, substituting the socket binding name and protocol with the ones you need.

```
[standalone@localhost:9999 /] /subsystem=web/connector=ajp:add(socket-binding=ajp, protocol="AJP/1.3", enabled=true, scheme="http")
```

[Report a bug](#)

13.2.12. Use TCP Communication for the Clustering Subsystem

By default, cluster nodes monitor each other's status using the UDP protocol. Some networks only allow TCP to be used. In this situation, you can add the **TCPPING** protocol stack to your configuration and use it as the default mechanism. These configuration options are available in the command-line based Management CLI.

The **mod_cluster** subsystem also uses UDP communication by default, and you can choose to use TCP here as well.

Refer to the following two procedures to configure JGroups and mod_cluster subsystems to use TCP for network communication:

- ▶ [Section 13.2.13, “Configure the JGroups Subsystem to Use TCP”](#)
- ▶ [Section 13.2.14, “Configure the mod_cluster Subsystem to Use TCP”](#)

[Report a bug](#)

13.2.13. Configure the JGroups Subsystem to Use TCP

The **mod_cluster** subsystem relies upon the JGroups subsystem to manage and track nodes leaving, joining, and failing over in the cluster. By default, the JGroups subsystem communicates using multicast UDP. Use the following procedure to configure the JGroups subsystem to use unicast TCP instead.

To configure the `mod_cluster` subsystem to use TCP as well, refer to [Section 13.2.14, “Configure the mod_cluster Subsystem to Use TCP”](#).

1. Run the Management CLI.

Launch the Management CLI, using the `EAP_HOME/bin/jboss-cli.sh` command in Linux or the `EAP_HOME\bin\jboss-cli.bat` command in Microsoft Windows Server. Type `connect` to connect to the domain controller on the localhost, or `connect IP_ADDRESS` to connect to a domain controller on a remote server.

2. Modify the following script to suit your environment.

Copy the following script into a text editor. If you use a different profile on a managed domain, change the profile name. If you use a standalone server, remove the `/profile=full-ha` portion of the commands. Modify the properties listed at the bottom of the command as follows. Each of these properties is optional.

`initial_hosts`

A comma-separated list of the hosts which are considered well-known, and will be available to look up the initial membership.

`port_range`

If desired, you can assign a port range. If you assign a port range of 2, and the initial port is 7600, then TCPPING will attempt to contact each host on ports 7600-7601. This property is optional.

`timeout`

An optional timeout value, in milliseconds, for cluster members.

`num_initial_members`

The number of nodes before the cluster is considered to be complete. This property is optional.

```
cd /profile=full-ha/subsystem=jgroups
./stack=tcping:add
cd stack=tcping
./transport=TRANSPORT:add(type=TCP,socket-binding=jgroups-tcp)
:add-protocol(type=TCPPING)
:add-protocol(type=MERGE2)
:add-protocol(type=FD_SOCK,socket-binding=jgroups-tcp-fd)
:add-protocol(type=FD)
:add-protocol(type=VERIFY_SUSPECT)
:add-protocol(type=BARRIER)
:add-protocol(type=pbcast.NAKACK)
:add-protocol(type=UNICAST2)
:add-protocol(type=pbcast.STABLE)
:add-protocol(type=pbcast.GMS)
:add-protocol(type=UFC)
:add-protocol(type=MFC)
:add-protocol(type=FRAG2)
:add-protocol(type=RSVP)
cd protocol=TCPPING
./property=initial_hosts/:add(value="HostA[7600],HostB[7600]")
./property=port_range/:add(value=0)
./property=timeout/:add(value=3000)
./property=num_initial_members/:add(value=3)
cd ../..
:write-attribute(name=default-stack,value=tcping)
```

3. Run the script in batch mode.



Warning

The servers running the profile have to be shutdown before executing the batch file.

At the Management CLI prompt, type `batch` and press the `Enter` key. The prompt changes to include a hash (#) symbol to indicate that you are in batch mode. This allows you to enter a series of commands. If any one of them fails, the entire operation will be rolled back.

Paste the modified script from the previous step, adding an extra newline at the end. Type `run-batch` to run the batch. After all commands have run, the message **The batch executed**

successfully appears.

Result

The **TCPING** stack is now available to the JGroups subsystem. If it is used, the JGroups subsystem uses TCP for all network communication. To configure the **mod_cluster** subsystem to use TCP as well, refer to [Section 13.2.14, “Configure the mod_cluster Subsystem to Use TCP”](#).

[Report a bug](#)

13.2.14. Configure the mod_cluster Subsystem to Use TCP

The **mod_cluster** subsystem uses multicast UDP for its network communication by default. If you wish, you can use unicast TCP communication instead. Use the following procedure to configure this behavior.

The **mod_cluster** subsystem relies upon JGroups to manage cluster nodes and failover behaviors. You can configure the JGroups subsystem to use TCP as well. Refer to [Section 13.2.13, “Configure the JGroups Subsystem to Use TCP”](#).

Procedure 13.6.

1. Modify the HTTPD configuration.

Modify the HTTPD configuration to disable server advertising and to use a proxy list instead. The proxy list is configured on the worker, and contains all of the **mod_cluster**-enabled HTTPD servers the worker can talk to.

The **mod_cluster** configuration for the HTTPD server is typically located in `/etc/httpd/` or the `etc/httpd/` directory within the HTTPD installation, if it is installed in a non-standard location. Refer to [Section 13.3.3, “Install the mod_cluster Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#) and [Section 13.3.4, “Configure Server Advertisement Properties for Your mod_cluster-enabled HTTPD”](#) for more information about the file itself. Open the file containing the virtual host which listens for MCPM requests (using the `EnableMCPMReceive` directive), and disable server advertising by changing the `ServerAdvertise` directive as follows.

```
ServerAdvertise Off
```

2. Disable advertising within the mod_cluster subsystem of JBoss Enterprise Application Platform, and provide a list of proxies.

You can disable advertising for the **mod_cluster** subsystem and provide a list of proxies, by using the web-based Management Console or the command-line Management CLI. The list of proxies is necessary because the **mod_cluster** subsystem will not be able to automatically discover proxies if advertising is disabled.

A. Management Console

- If you use a managed domain, you can only configure **mod_cluster** in profiles where it is enabled, such as the **ha** and **full-ha** profiles.
- Log in to the Management Console and select the **Profiles** label at the top right of the screen. If you use a managed domain, select either the **ha** or **full-ha** profile from the **Profiles** selection box at the top left of the **Profiles** page.
- Click the **Subsystems** menu to expand it. Expand the **Web** sub-menu, and select **Modcluster**.
- Click the **Edit** button at the top, to edit the options which pertain to the entire **mod_cluster** subsystem. Change the value of **Advertise** to **false**. Use the **Save** button to save the settings.
- Click the tab labeled **Proxies** near the bottom of the screen. Click the **Edit** button in the **Proxies** sub-page, and enter a list of proxy servers. The correct syntax is a comma-separated list of **HOSTNAME:PORT** strings, like the following.

```
10.33.144.3:6666,10.33.144.1:6666
```

Click the **Save** button to save your changes.

B. Management CLI

The following two Management CLI commands create the same configuration as the Management Console instructions above. They assume that you run a managed domain and that your server group uses the **full-ha** profile. If you use a different profile, change its name in the commands. If you use a standalone server using the **standalone-ha** profile, remove the `/profile=full-ha` portion of the commands.

```
/profile=full-ha/subsystem=modcluster/mod-cluster-
config=configuration/:write-attribute(name=advertise,value=false)

/profile=full-ha/subsystem=modcluster/mod-cluster-
config=configuration/:write-attribute(name=proxy-
list,value="10.33.144.3:6666,10.33.144.1:6666")
```

Result

The TCP protocol is used for **mod_cluster** communication by default.

[Report a bug](#)

13.3. Web, HTTP Connectors, and HTTP Clustering

13.3.1. About the mod_cluster HTTP Connector

mod_cluster is the module which enables load balancing in the JBoss Web container. It is referred to as a *connector*. Which connector you use depends on which web container you choose to use with the JBoss Enterprise Application Platform. To learn about other connectors, refer to one of the following:

- ▶ [Section 13.4.1, “About the Apache mod_jk HTTP Connector”](#)
- ▶ [Section 13.6.1, “About the Internet Server API \(ISAPI\) HTTP Connector”](#)
- ▶ [Section 13.7.1, “About the Netscape Server API \(NSAPI\) HTTP Connector”](#)

The *mod_cluster* connector has several advantages.

- ▶ *mod_cluster Management Protocol (MCMP)* is an additional connection between the application server nodes and httpd, used by application server nodes to transmit server-side load balance factors and lifecycle events back to the web container via a custom set of HTTP methods.
- ▶ Dynamic configuration of HTTPD proxies allows the JBoss Enterprise Application Platform to adapt on the fly, with no additional configuration.
- ▶ The application servers perform the load-balancing factor calculations. This makes load balancing metrics more accurate than other connectors.
- ▶ *mod_cluster* gives fine-grained application lifecycle control. Each server forwards any web application context lifecycle events to the proxy, informing it to start or stop routing requests for a given context in the server. This prevents end users from seeing 404 errors due to unavailable resources.
- ▶ AJP is optional. Apache HTTP requires the use of AJP, but *mod_cluster* can use HTTP, HTTPS, or AJP.

[Report a bug](#)

13.3.2. Configure the mod_cluster Subsystem

In the web-based Management Console, the **mod_cluster** options are available as part of the Web subsystem configuration area. Click the **Profiles** tab at the top left. If you use a managed domain, select the correct profile to configure from the **Profile** selection box at the upper right. By default, the **ha** and **full-ha** profiles have the **mod_cluster** subsystem enabled. If you use a standalone server, you need to use the **standalone-ha** profile to start the server. Click the **Web** item in the left-hand menu, and choose **Modcluster** from the sub-menu. The options are explained in the tables below. Overall configuration is shown first, followed by configuration of sessions, web contexts, proxies, SSL, and Networking. Each of these has its own tab within the **Modcluster** configuration screen.

 **Note**

The **Modcluster** configuration page is only visible for profiles with the HA Clustering subsystem enabled. These profiles are **ha** and **full-ha** for a managed domain, or **standalone-ha** for a standalone server.

Table 13.7. mod_cluster Configuration Options

Option	Description	CLI Command
Load Balancing Group	If this is not null, requests are sent to a specific load balancing group on the load balancer. Leave this blank if you do not want to use load balancing groups. This is unset by default.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=load- balancing- group,value=myGroup)</pre>
Balancer	The name of the balancer. This should match the configuration of the HTTPD proxy.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=balancer ,value=myBalancer)</pre>
Advertise Socket	The name of the socket binding to use for cluster advertising.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=advertis e- socket,value=modcluster)</pre>
Advertise Key	A string containing the security key for advertising.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=advertis e-security- key,value=myKey)</pre>
Advertise	Whether or not advertising is enabled. Defaults to true .	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=advertis e,value=true)</pre>

Table 13.8. mod_cluster Session Configuration Options

Option	Description	CLI Command
Sticky Session	Whether to use sticky sessions for requests. This means that after the client makes a connection to a specific cluster node, further communication is routed to that same node unless it becomes unavailable. This defaults to true , which is the recommended setting.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=sticky- session,value=true)</pre>
Sticky Session Force	If true , a request is not redirected to a new cluster node if its initial node becomes unavailable. Instead, it fails. This defaults to false .	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=sticky- session- force,value=false)</pre>
Sticky Session Remove	Remove session information on failover. Disabled by default.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=sticky- session- remove,value=false)</pre>

Table 13.9. mod_cluster Web Context Configuration Options

Option	Description	CLI Command
Auto Enable Contexts	Whether to add new contexts to mod_cluster by default or not. This defaults to true . If you change the default and need to enable context manually, the Web Application can enable its context using the enable() MBean method, or via the mod_cluster manager, which is a web application which runs on the HTTPD proxy on a named virtual host or port which is specified in that HTTPD's configuration.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=auto- enable- contexts,value=true)</pre>
Excluded Contexts	A comma-separated list of contexts that mod_cluster should ignore. If no host is indicated, the host is assumed to be localhost . ROOT indicates the root context of the Web Application. The default value is ROOT,invoker,jbossws,juddi,console .	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=excluded- contexts,value="ROOT,in voker,jbossws,juddi,con sole")</pre>

Table 13.10. mod_cluster Proxy Configuration Options

Option	Description	CLI Command
Proxy URL	If defined, this value will be prepended to the URL of MCMP commands.	/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-url,value=myhost)
Proxy List	A comma-separated list of HTTPD proxy addresses, in the format hostname:port . This indicates the list of proxies that the mod_cluster process will attempt to communicate with initially.	/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-list,value="127.0.0.1,127.0.0.2")

Configure SSL Communication for mod_cluster

By default, **mod_cluster** communication happens over an unencrypted HTTP link. If you set the connector scheme to **HTTPS** (refer to [Table 13.8, “mod_cluster Session Configuration Options”](#)), the settings below tell **mod_cluster** where to find the information to encrypt the connection.

Table 13.11. mod_cluster SSL Configuration Options

Option	Description	CLI Command
Key Alias	The key alias, which was chosen when the certificate was created.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=key- alias,value=jboss)</pre>
Password	The password, which was chosen when the certificate was created.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=password ,value=changeit)</pre>
Cert File	The location of the certificate file.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=ca- certificate- file,value=\${user.home} /jboss.crt)</pre>
Key File	The location of the key file for the certificate.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=certific ate-key- file,value=\${user.home} /.keystore)</pre>
Cipher Suite	The allowed encryption cipher suite.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=cipher- suite,value=ALL)</pre>
Revocation URL	The URL of the Certificate Authority revocation list.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=ca- revocation- url,value=jboss.crl)</pre>
Protocol	The SSL protocols which are enabled.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/ss l=configuration/:write- attribute(name=protocol ,value=SSLv3)</pre>

Configure mod_cluster Networking Options

The available **mod_cluster** networking options control several different timeout behaviors for different types of services that the **mod_cluster** service communicates with.

Table 13.12. mod_cluster Networking Configuration Options

Option	Description	CLI Command
Node Timeout	Timeout (in seconds) for proxy connections to a node. That is the time mod_cluster will wait for the back-end response before returning error. That corresponds to timeout in the worker mod_proxy documentation. A value of -1 indicates no timeout. Note that mod_cluster always uses a cping/cpong before forwarding a request and the connectiontimeout value used by mod_cluster is the ping value.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=node- timeout,value=-1)</pre>
Socket Timeout	Number of milliseconds to wait for a response from an httpd proxy to MCMP commands before timing out, and flagging the proxy as in error.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=socket- timeout,value=20)</pre>
Stop Context Timeout	The amount of time, measure in units specified by stopContextTimeoutUnit , for which to wait for clean shutdown of a context (completion of pending requests for a distributable context; or destruction/expiration of active sessions for a non-distributable context).	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=stop- context- timeout,value=10)</pre>
Max Attempts	Number of times an HTTPD proxy will attempt to send a given request to a worker before giving up. The minimum value is 1 , meaning try only once. The mod_proxy default is also 1 , which means that no retry occurs.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=max- attempts,value=1)</pre>
Flush Packets	Whether or not to enable packet flushing to the HTTPD server. Defaults to false .	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=flush- packets,value=false)</pre>
Flush Wait	How long, in seconds, to wait before flushing packets to the HTTPD server. Defaults to -1 . A value of -1 means to wait forever before flushing packets.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/:w rite- attribute(name=flush- wait,value=-1)</pre>
Ping	How long, in seconds, to wait for	

	<p>How long, in seconds, to wait for a response to a ping from a cluster node. Defaults to 10 seconds.</p>	<code>/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ping,value=10)</code>
TTL	<p>Time to live (in seconds) for idle connections above smax, default is 60</p> <p>When nodeTimeout is not defined the ProxyTimeout directive Proxy is used. If ProxyTimeout is not defined the server timeout Timeout is used. This defaults to 300 seconds. nodeTimeout, ProxyTimeout, are Timeout are set at the socket level.</p>	<code>/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ttl,value=-1)</code>
Worker Timeout	<p>How long, in seconds, to wait for an available worker process from the external HTTPD server to process a request. Defaults to -1, which means that Modcluster waits indefinitely for the HTTPD worker to process the request.</p>	<code>/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=worker-timeout,value=-1)</code>

mod_cluster Load Provider Configuration Options

The following **mod_cluster** configuration options are not available in the web-based Management Console, but can only be set using the command-line Management CLI.

The simple load processor is used if no dynamic load processor is present. It gives each cluster member a load factor of **1**, and distributes work evenly without taking any sort of load balancing algorithm into account. To add it, use the following CLI command: **/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/simple-load-provider:add**

A dynamic load provider can be configured to use a variety of different algorithms, in combination, to determine which cluster node receives the next request. The default dynamic load provider uses **busyness** as the determining factor. The list of possible factors is shown below. You can create your own load provider to suit your own environment, as well. The following options of the dynamic load provider can be changed..

Table 13.13. mod_cluster Dynamic Load Provider Options

Option	Description	CLI Command
Decay	The factor by which historical metrics should decay in significance.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/dy- namic-load- provider=configuration/ :write- attribute(name=decay, va- lue=2)</pre>
History	The number of historic load metric records to consider when determining the load.	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/dy- namic-load- provider=configuration/ :write- attribute(name=history, value=9)</pre>
Load Metric	The only load metric included in the dynamic load provider is busyness , which tries to send each new request to the least busy worker. You can set the capacity of your worker (1 means 100% capacity) and the weight assigned to the busyness metric overall. .	<pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/dy- namic-load- provider=configuration/ load- metric=busyness/:write- attribute(name=capacity ,value=1.0)</pre> <pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/dy- namic-load- provider=configuration/ load- metric=busyness/:write- attribute(name=type, val- ue=busyness)</pre> <pre>/profile=full- ha/subsystem=modcluster /mod-cluster- config=configuration/dy- namic-load- provider=configuration/ load- metric=busyness/:write- attribute(name=weight, v- alue=1)</pre>

Load Metric Algorithms

cpu

The cpu load metric uses average CPU load to determine which cluster node receives the next work load.

mem

The mem load metric uses free native RAM as a load factor. Usage of this metric is discouraged because it provides a value that includes buffers and cache, so it is always a very

low figure on every decent system with a good memory management.

heap

The heap load metric uses the heap usage to determine which cluster receives the next work load.

sessions

The session load metric uses the number of active sessions as a metric.

requests

The requests load metric uses the number of client requests to determine which cluster node receives the next work load. For instance, capacity 1000 means that 1000 requests/sec is considered to be a "full load".

send-traffic

The send-traffic load metric uses the amount of traffic sent from the worker node to the clients. E.g. the default capacity of 512 indicates that the node should be considered under full load if the average outbound traffic is 512 KB/s or higher.

receive-traffic

The receive-traffic load metric uses the amount of traffic sent to the worker node from the clients. E.g. the default capacity of 1024 indicates that the node should be considered under full load if the average inbound traffic is 1024 KB/s or higher.

busyness

This metric represents the amount of threads from the thread pool being busy serving requests.

Example 13.5. Set a Load Balancer Metric

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/dynamic-
load-provider=configuration/load-metric=cpu/:write-
attribute(name="weight",value="3")
```

[Report a bug](#)

13.3.3. Install the mod_cluster Module Into Apache HTTPD or Enterprise Web Server HTTPD

Prerequisites

- ▶ To perform this task, you must be using Apache HTTPD installed in Red Hat Enterprise Linux 6, or JBoss Enterprise Web Server, or the stand-alone HTTPD included as a separate downloadable component of JBoss Enterprise Application Platform 6.
- ▶ If you need to install Apache HTTPD in Red Hat Enterprise Linux 6, use the instructions from the *Red Hat Enterprise Linux 6 Deployment Guide*, available from <https://access.redhat.com/knowledge/docs/>.
- ▶ If you need to install the stand-alone HTTPD included as a separate downloadable component of JBoss Enterprise Application Platform 6, refer to [Section 13.2.9, “Install the Apache HTTPD included with JBoss Enterprise Application Platform 6”](#).
- ▶ If you need to install JBoss Enterprise Web Server, use the instructions from the *JBoss Enterprise Web Server Installation Guide*, available from <https://access.redhat.com/knowledge/docs/>.
- ▶ Download the **Webserver Connector Natives** package for your operating system and architecture from the Red Hat Customer Portal at <https://access.redhat.com>. This package contains the mod_cluster binary HTTPD modules precompiled for your operating system. After you extract the archive, the modules are located in the **modules/native/lib/httpd/modules/** directory. The **etc/** contains some example configuration files, and the **share/** directory contains some supplemental documentation.
- ▶ You must be logged in with administrative (root) privileges.

Procedure 13.7. Task

1. Determine your HTTPD configuration location.

Your HTTPD configuration location will be different depending on whether you are using Red Hat Enterprise Linux's Apache HTTPD, the stand-alone HTTPD included as a separate downloadable component with JBoss Enterprise Application Platform 6, or the HTTPD available in JBoss Enterprise Web Server. It is one of the following three options, and is referred to in the rest of this task as **HTTPD_HOME**.

- » Apache HTTPD - `/etc/httpd/`
- » JBoss Enterprise Application Platform HTTPD - This location is chosen by you, based on the requirements of your infrastructure.
- » JBoss Enterprise Web Server HTTPD - `EWS_HOME/httpd/`

2. Copy the modules to the HTTPD modules directory.

Copy the four modules (the files ending in `.so`) from the `modules/native/lib/httpd/modules/` directory of the extracted Webserver Natives archive to the `HTTPD_HOME/modules/` directory.

3. For Enterprise Web Server, disable the mod_proxy_balancer module.

If you use JBoss Enterprise Web Server, the `mod_proxy_balancer` module is enabled by default. It is incompatible with `mod_cluster`. To disable it, edit the `HTTPD_HOME/conf/httpd.conf` and comment out the following line by placing a # (hash) symbol before the line which loads the module. The line is shown without the comment and then with it, below.

```
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

```
# LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

Save and close the file.

4. Configure the mod_cluster module.

- a. Open `HTTPD_HOME/conf/httpd.conf` in a text editor and add the following to the end of the file:

```
# Include mod_cluster's specific configuration file
Include conf/JBoss_HTTP.conf
```

Save and exit the file.

- b. Create a new file called `HTTPD_HOME/httpd/conf/JBoss_HTTP.conf` and add the following to it.

```
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so
```

This causes Apache HTTPD to automatically load the modules that `mod_cluster` needs in order to function.

5. Create a proxy server listener.

Continue editing `HTTPD_HOME/httpd/conf/JBoss_HTTP.conf` and add the following minimal configuration, replacing the values in capital letters with sensible ones for your system.

```

Listen IP_ADDRESS:PORT
<VirtualHost IP_ADDRESS:PORT>
    <Location />
        Order deny,allow
        Deny from all
        Allow from *.MYDOMAIN.COM
    </Location>

    KeepAliveTimeout 60
    MaxKeepAliveRequests 0
    EnableMCPMReceive On

    ManagerBalancerName mycluster
    ServerAdvertise On

</VirtualHost>

```

These directives create a new virtual server which listens on **IP_ADDRESS:PORT**, allows connections from **MYDOMAIN.COM**, and advertises itself as a balancer called **mycluster**. These directives are covered in detail in the documentation for Apache Web Server. To learn more about the **ServerAdvertise** and **EnableMCPMReceive** directives, and the implications of server advertisement, refer to [Section 13.3.4, “Configure Server Advertisement Properties for Your mod_cluster-enabled HTTPD”](#).

Save the file and exit.

6. Restart the HTTPD.

The way to restart the HTTPD depends on whether you are using Red Hat Enterprise Linux's Apache HTTPD or the HTTPD included in JBoss Enterprise Web Server. Choose one of the two methods below.

A. Red Hat Enterprise Linux 6 Apache HTTPD

Issue the following command:

```
[root@host]# service httpd restart
```

B. JBoss Enterprise Web Server HTTPD

JBoss Enterprise Web Server runs on both Red Hat Enterprise Linux and Microsoft Windows Server. The method for restarting the HTTPD is different for each.

A. Red Hat Enterprise Linux

In Red Hat Enterprise Linux, Enterprise Web Server installs its HTTPD as a service. To restart the HTTPD, issue the following two commands:

```
[root@host ~]# service httpd stop
[4root@host ~]# service httpd start
```

B. Microsoft Windows Server

Issue the following commands in a command prompt with administrative privileges:

```
C:\> net stop httpd
C:\> net start httpd
```

Result

The Apache HTTPD is now configured as a load balancer, and can work with the **mod_cluster** subsystem running JBoss Enterprise Application Platform 6. To configure the JBoss Enterprise Application Platform to be aware of mod_cluster, refer to [Section 13.3.5, “Configure a mod_cluster Worker Node”](#).

[Report a bug](#)

13.3.4. Configure Server Advertisement Properties for Your mod_cluster-enabled HTTPD

Overview

For instructions on configuring your HTTPD to interact with the mod_cluster load balancer, refer to [Section 13.3.3, “Install the mod_cluster Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#). One aspect of the configuration which needs more explanation is *server advertisement*.

When server advertisement is active, the **HTTPD** broadcasts messages containing the IP address and port number specified in the `mod_cluster` virtual host. To configure these values, refer to [Section 13.3.3, “Install the mod_cluster Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#). If UDP multicast is not available on your network, or you prefer to configure worker nodes with a static list of proxy servers, you can disable server advertisement and manually configure the worker nodes. Refer to [Section 13.3.5, “Configure a mod_cluster Worker Node”](#) for information on configuring a worker node.

The changes in this procedure need to be made to the `httpd.conf` associated with your Apache HTTPD instance. This is often `/etc/httpd/conf/httpd.conf` in Red Hat Enterprise Linux, or may be in the `etc/` directory of your stand-alone Apache HTTPD instance.

Procedure 13.8. Task

- 1. Disable the AdvertiseFrequency parameter, if it exists.**

If you have a line like the following in your `<VirtualHost>` statement, comment it out by putting a `#` (hash) character before the first character. The value may be different from **5**.

```
AdvertiseFrequency 5
```

- 2. Add the directive to disable server advertisement.**

Add the following directive inside the `<VirtualHost>` statement, to disable server advertisement.

```
ServerAdvertise Off
```

- 3. Enable the ability to receive MCPM messages.**

Add the following directive to allow the **HTTPD** server to receive MCPM messages from the worker nodes.

```
EnableMCPMReceive On
```

- 4. Restart the **HTTPD** server.**

Restart the **HTTPD** server by issuing one of the following, depending on whether you use Red Hat Enterprise Linux or Microsoft Windows Server.

A. **Red Hat Enterprise Linux**

```
[root@host ]# service httpd restart
```

B. **Microsoft Windows Server**

```
C:\> net service http
C:\> net service httpd start
```

Result

The **HTTPD** no longer advertises the IP address and port of your `mod_cluster` proxy. To reiterate, you need to configure your worker nodes to use a static address and port to communicate with the proxy. Refer to [Section 13.3.5, “Configure a mod_cluster Worker Node”](#) for more details.

[Report a bug](#)

13.3.5. Configure a mod_cluster Worker Node

Configure a mod_cluster worker node

A `mod_cluster` worker node consists of an JBoss Enterprise Application Platform server. This server can be part of a server group in a Managed Domain, or a standalone server. A separate process runs within JBoss Enterprise Application Platform, which manages all of the nodes of the cluster. This is called the master. For more conceptual information about worker nodes, refer to [Section 13.1.4, “Worker Node”](#). For an overview of **HTTPD** load balancing, refer to [Section 13.1.3, “Overview of HTTP Connectors”](#).

The master is only configured once, via the `mod_cluster` subsystem. To configure the `mod_cluster` subsystem, refer to [Section 13.3.2, “Configure the mod_cluster Subsystem”](#). Each worker node is configured separately, so repeat this procedure for each node you wish to add to the cluster.

If you use a managed domain, each server in a server group is a worker node which shares an identical configuration. Therefore, configuration is done to an entire server group. In a standalone server,

configuration is done to a single JBoss JBoss Enterprise Application Platform instance. The configuration steps are otherwise identical.

Worker Node Configuration

- » If you use a standalone server, it must be started with the **standalone-ha** profile.
- » If you use a managed domain, your server group must use the **ha** or **full-ha** profile, and the **ha-sockets** or **full-ha-sockets** socket binding group. JBoss Enterprise Application Platform ships with a cluster-enabled server group called **other-server-group** which meets these requirements.



Note

Where Management CLI commands are given, they assume you use a managed domain. If you use a standalone server, remove the **/profile=full-ha** portion of the commands.

Procedure 13.9. Configure a Worker Node

1. Configure the network interfaces.

By default, the network interfaces all default to **127.0.0.1**. Every physical host which 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.

To change the IP address of a JBoss Enterprise Application Platform host, you need to shut it down and edit its configuration file directly. This is because the Management API which drives the Management Console and Management CLI relies on a stable management address.

Follow these steps to change the IP address on each server in your cluster to the master's public IP address.

- a. Shut down the server completely.
- b. Edit either the **host.xml**, which is in **EAP_HOME/domain/configuration/** for a managed domain, or the **standalone-ha.xml** file, which is in **EAP_HOME/standalone/configuration/** for a standalone server.
- c. Locate the **<interfaces>** element. Three interfaces are configured, **management**, **public**, and **unsecured**. For each of these, change the value **127.0.0.1** to the external IP address of the host.
- d. For hosts that participate in a managed domain but are not the master, locate the **<host>** element. Note that it does not have the closing **>** symbol, because it contains attributes. Change the value of its name attribute from **master** to a unique name, a different one per slave. This name will also be used for the slave to identify to the cluster, so make a note of it.
- e. For newly-configured hosts which need to join a managed domain, find the **<domain-controller>** element. Comment out or remove the **<local />** element, and add the following line, changing the IP address (**X.X.X.X**) to the address of the domain controller. This step does not apply for a standalone server.

```
<remote host="X.X.X.X" port="${jboss.domain.master.port:9999}" security-realm="ManagementRealm"/>
```

- f. Save the file and exit.

2. Configure authentication for each slave server.

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/add-user.sh** command. Add a user with the same username as the slave, to the **ManagementRealm**. When asked if this user will need to authenticate to an external JBoss AS instance, answer **yes**. An example of the input and output of the command is below, for a slave called **slave1**, with password **changeme**.

```

user:bin user$ ./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.
Realm (ManagementRealm) :
Username : slave1
Password : changeme
Re-enter Password : changeme
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave1' to file '/home/user/jboss-eap-
6.0/standalone/configuration/mgmt-users.properties'
Added user 'slave1' to file '/home/user/jboss-eap-
6.0/domain/configuration/mgmt-users.properties'
Is this new user going to be used for one AS process to connect to another AS
process e.g. slave domain controller?
yes/no? yes
To represent the user add the following to the server-identities definition
<secret value="Y2hhbmdlbWU=" />

```

3. Copy the <secret> element from the add-user.sh output.

Copy the value from the last line of the `add-user.sh` output. You need to add this value to your slave's configuration file in the next step.

4. Modify the slave host's security realm to use the new authentication.

Re-open the slave host's `host.xml` or `standalone-ha.xml` file and locate the `<security-realms>` element. Add the following block of XML code directly below the `<security-realm name="ManagementRealm">` line, replacing the `<secret value="Y2hhbmdlbWU=" />` line with the one from the previous step.

```

<server-identities>
  <secret value="Y2hhbmdlbWU=" />
</server-identities>

```

Save and exit the file.

5. 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.

Result

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 HTTPD server or load balancer. Each node of the cluster discovers the other nodes using automatic discovery, by default. To configure automatic discovery, and the other specific settings of the `mod_cluster` subsystem, refer to [Section 13.3.2, “Configure the mod_cluster Subsystem”](#). To configure the Apache HTTPD server, refer to [Section 13.2.10, “Use an External HTTPD as the Web Front-end for JBoss Enterprise Application Platform Applications”](#).

[Report a bug](#)

13.4. Apache mod_jk

13.4.1. About the Apache mod_jk HTTP Connector

`mod_jk` provides HA clustering if you use the Apache Tomcat container instead of the built-in JBoss Web container. The `mod_jk` connector is maintained by Apache. Its documentation is located at <http://tomcat.apache.org/connectors-doc/>.

The JBoss Enterprise Application Platform can accept work loads from an Apache HTTPD proxy server. The proxy server accepts client requests from the web front-end, and passes the work to participating JBoss Enterprise Application Platform servers. If sticky sessions are enabled, the same client request always goes to the same JBoss Enterprise Application Platform server, unless the server is unavailable.

Unlike the mod_cluster JBoss HTTP connector, an Apache proxy server does not know the status of deployments on servers or server groups, and cannot adapt where it sends its work accordingly.

The advantages of Apache mod_jk are that it communicates over the AJP 1.3 protocol, and that it is available for Red Hat Enterprise Linux 5 and Microsoft Windows servers without a JBoss Enterprise Web Server entitlement.

Unless you need to use the AJP protocol or you need to run Red Hat Enterprise Linux 5 operating system, mod_cluster is the recommended load balancer because it integrates tightly with the web server and the JBoss Enterprise Application Platform. mod_cluster runs on Red Hat Enterprise Linux 6 and Microsoft Windows servers, over the HTTP and HTTPS protocols. For more information on mod_cluster, refer to [Section 13.3.1, “About the mod_cluster HTTP Connector”](#).

Next step: Configure the JBoss Enterprise Application Platform to participate in a mod_jk load balancing group

- ▶ [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#)
- ▶ [Section 13.4.3, “Install the Mod_jk Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#)

[Report a bug](#)

13.4.2. Configure the JBoss Enterprise Application Platform to Communicate with Apache Mod_jk

Overview

The mod_jk HTTP connector has a single component, the **mod_jk.so** module loaded by the HTTPD. This module receives client requests and forwards them to the container, in this case the JBoss Enterprise Application Platform. The JBoss Enterprise Application Platform also needs to be configured to accept these requests and send replies back to the HTTPD.

Configuring the HTTPD is covered in [Section 13.4.3, “Install the Mod_jk Module Into Apache HTTPD or Enterprise Web Server HTTPD”](#).

In order for JBoss Enterprise Application Platform to be able to communicate with the Apache HTTPD, it needs to have the AJP/1.3 HTTPD connector enabled. This connector is present by default in the following configurations:

- ▶ In a managed domain, in server groups using the **ha** and **full-ha** profiles, and the **ha** or **full-ha** socket binding group. The **other-server-group** server group is configured correctly in a default installation.
- ▶ In a standalone server, the **standalone-ha** profile is provided for clustered configurations. To start the standalone server with this profile, issue the following command, from the **EAP_HOME/bin** directory.

```
[user@host bin]$ ./bin/standalone.sh --server-config=standalone-ha.xml
```

[Report a bug](#)

13.4.3. Install the Mod_jk Module Into Apache HTTPD or Enterprise Web Server HTTPD

Prerequisites

- ▶ To perform this task, you must be using Apache HTTPD installed in Red Hat Enterprise Linux, or the HTTPD installed in JBoss Enterprise Web Server.
- ▶ If you need to install Apache HTTPD, use the instructions from the *Red Hat Enterprise Linux Deployment Guide*, available from <https://access.redhat.com/knowledge/docs/>.
- ▶ If you need to install JBoss Enterprise Web Server, use the instructions from the *JBoss Enterprise Web Server Installation Guide*, available from <https://access.redhat.com/knowledge/docs/>.
- ▶ If you are using Apache HTTPD, download the JBoss Enterprise Application Platform Native Components package for Red Hat Enterprise Linux from the Red Hat Customer Service Portal at <https://access.redhat.com>. This package contains both the mod_jk and mod_cluster binaries precompiled for Red Hat Enterprise Linux. If you are using JBoss Enterprise Web Server, it already includes the binary for mod_jk.
- ▶ You must be logged in with administrative (root) privileges.

Procedure 13.10. Task

1. Determine your HTTPD configuration location.

Your HTTPD configuration location will be different depending on whether you are using Red Hat Enterprise Linux's Apache HTTPD or the HTTPD available in JBoss Enterprise Web Platform. It is one of the following two options, and is referred to in the rest of this task as **HTTPD_HOME**.

- » Apache HTTPD - `/etc/httpd/`
- » JBoss Enterprise Web Server HTTPD - `EWS_HOME/httpd/`

2. Configure the mod_jk module.

- a. Open `HTTPD_HOME/conf/httpd.conf` in a text editor and add the following to the end of the file:

```
# Include mod_jk's specific configuration file
Include conf/mod-jk.conf
```

- b. Create a new file called `HTTPD_HOME/etc/httpd/conf/mod-jk.conf` and add the following to it:



The JkMount directive

The **JkMount** directive specifies which URLs Apache should forward to the mod_jk module. Based on the directive's configuration, mod_jk forwards the received URL to the correct Servlet containers.

To serve static content directly, and only use the load balancer for Java applications, the URL path should be `/application/*`. To use mod_jk as a load balancer, use the value `/*`, to forward all URLs to mod_jk.

```
# 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/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 +ForwardURICcompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
# The default setting only sends Java application data to mod_jk.
# Use the commented-out line to send all URLs through mod_jk.
# JkMount /* loadbalancer
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
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

Look over the values and make sure they are reasonable for your set-up. When you are satisfied, save the file.

c. 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.

- Add the following to the **HTTPD_HOME/conf/mod-jk.conf** file:

```
# You can 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/uriworkermap.properties
```

- Create a new file called **HTTPD_HOME/conf/uriworkermap.properties**, with a line for each URL pattern to be matched. The following example shows examples of the syntax of the file.

```
# Simple worker configuration file
/*=loadbalancer
```

d. Copy the mod_jk.so file to the HTTPD's modules directory



Note

This is only necessary if your HTTPD does not have **mod_jk.so** in its **modules/** directory. You can skip this step if you are using the Apache HTTPD server included as a download as part of JBoss Enterprise Application Platform 6.

Extract the Native Components ZIP package. Locate the **mod_jk.so** file in either the **native/lib/httpd/modules/** or **native/lib64/httpd/modules/** directory, depending on whether your operating system is 32-bit or 64-bit.

Copy the file to the **HTTPD_HOME/modules/** directory.

3. Configure the mod_jk worker nodes.

- Create a new file called **HTTPD_HOME/conf/workers.properties**. Use the following example as your starting point, and modify the file to suit 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 a detailed description of the syntax of the **workers.properties** file, and advanced configuration options, refer to [Section 13.4.4, “Configuration Reference for Apache Mod_jk Workers”](#).

4. Restart the HTTPD.

The way to restart the HTTPD depends on whether you are using Red Hat Enterprise Linux's Apache HTTPD or the HTTPD included in JBoss Enterprise Web Server. Choose one of the two methods below.

A. Red Hat Enterprise Linux's Apache HTTPD

Issue the following command:

```
[root@host]# service httpd restart
```

B. JBoss Enterprise Web Server HTTPD

JBoss Enterprise Web Server runs on both Red Hat Enterprise Linux and Microsoft Windows Server. The method for restarting the HTTPD is different for each.

A. Red Hat Enterprise Linux

In Red Hat Enterprise Linux, Enterprise Web Server installs its HTTPD as a service. To restart the HTTPD, issue the following two commands:

```
[root@host ~]# service httpd stop  
[root@host ~]# service httpd start
```

B. Microsoft Windows Server

Issue the following commands in a command prompt with administrative privileges:

```
C:\> net stop httpd  
C:\> net start httpd
```

Result

The Apache HTTPD is now configured to use the mod_jk load balancer. To configure the JBoss Enterprise Application Platform to be aware of mod_jk, refer to [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#).

[Report a bug](#)

13.4.4. Configuration Reference for Apache Mod_jk Workers

The **workers.properties** file defines the behavior of the worker nodes which mod_jk passes client requests to. In Red Hat Enterprise Linux, the file resides in **/etc/httpd/conf/workers.properties**. The **workers.properties** file defines where the different servlet containers are located, and the way the work load should be balanced across them.

The configuration is divided into three sections. The first section deals with global properties, which apply to all worker nodes. The second section contains settings which apply to a specific worker. The third section contains settings which apply to a specific node balanced by the worker.

The general structure of a property is **worker.WORKER_NAME.DIRECTIVE**, where **WORKER_NAME** is a unique name for the worker, and **DIRECTIVE** is the setting to be applied to the worker.

Configuration reference for Apache mod_jk workers

Node templates specify default per-node settings. You can override the template within the node settings itself. You can see an example of node templates in [Example 13.6, “Example workers.properties file”](#).

Table 13.14. Global properties

Property	Description
worker.list	The list of worker names used by mod_jk. These workers are available to receive requests.

Table 13.15. Per-worker properties

Property	Description
type	The type of the worker. The default type is ajp13 . Other possible values are ajp14 , lb , status . For more information on these directives, refer to the Apache Tomcat Connector AJP Protocol Reference at http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html .
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 names. This is set per worker, not per node. It affects all nodes balanced by that worker type.
sticky_session	Specifies whether requests from the same session are always routed to the same worker. The default is 0 , meaning that sticky sessions are disabled. To enable sticky sessions, set it to 1 . Sticky sessions should usually be enabled, unless all of your requests are truly stateless. This is set per worker, not per node. It affects all nodes balanced by that worker type.

Table 13.16. Per-node properties

Property	Description
host	The hostname or IP address of the worker. The worker node must support the ajp protocol stack. The default value is localhost .
port	The port number of the remote 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.
ping_mode	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 . <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 value. 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_interv al	The timeout values for the connection probe settings above. The value is specified in milliseconds, and the default value for ping_timeout is 10000.
lbfactor	Specifies the load-balancing factor for an individual worker, and only applies to a member worker of a load balancer. This is useful to give a more powerful server more of the work load. To give a worker 3 times the default load, set this to 3:worker.my_worker.lbfactor=3

Example 13.6. Example workers.properties file

```

worker.list=ajp13,lb

worker.balancer1.sticky_sessions=1
worker.balancer1.balance_workers=node1
worker.balancer2.sticky_session=1
worker.balancer2.balance_workers=node2,node3

worker.nodetemplate.type=ajp13
worker.nodetemplate.port=8009

worker.node1.template=nodetemplate
worker.node1.host=localhost
worker.node1.ping_mode=CI
worker.node1.connection_ping_interval=9000
worker.node1.lbfactor=1

worker.node1.template=nodetemplate
worker.node2.host=192.168.1.1
worker.node2.ping_mode=A

worker.node1.template=nodetemplate
worker.node3.host=192.168.1.2

```

Further configuration details for Apache mod_jk are out of the scope of this document. Refer to the Apache documentation at <http://tomcat.apache.org/connectors-doc/> for further instructions.

[Report a bug](#)

13.5. Apache mod_proxy

13.5.1. About the Apache mod_proxy HTTP Connector

Apache provides two different proxying and load balancing modules for its HTTPD: **mod_proxy** and **mod_jk**. To learn more about **mod_jk**, refer to [Section 13.4.1, “About the Apache mod_jk HTTP Connector”](#). The JBoss Enterprise Application Platform supports use of either of these, although **mod_cluster**, the JBoss HTTP connector, more closely couples the JBoss Enterprise Application Platform and the external HTTPD, and is the recommended HTTP connector. Refer to [Section 13.1.3, “Overview of HTTP Connectors”](#) for an overview of all supported HTTP connectors, including advantages and disadvantages.

Unlike **mod_jk**, **mod_proxy** supports connections over HTTP and HTTPS protocols. Each of them also support the AJP protocol.

mod_proxy can be configured in standalone or load-balanced configurations, and it supports the notion of sticky sessions.

The **mod_proxy** module requires JBoss Enterprise Application Platform to have the HTTP or HTTPS web connector configured. This is part of the Web subsystem. Refer to [Section 13.2.2, “Configure the Web Subsystem”](#) for information on configuring the Web subsystem.

[Report a bug](#)

13.5.2. Install the Mod_proxy HTTP Connector Into Apache HTTPD

Overview

mod_proxy is a load-balancing module provided by Apache. This task presents a basic configuration. For more advanced configuration, or additional details, refer to Apache's **mod_proxy** documentation at https://httpd.apache.org/docs/2.2/mod/mod_proxy.html. For more details about **mod_proxy** from the perspective of the JBoss Enterprise Application Platform, refer to [Section 13.5.1, “About the Apache mod_proxy HTTP Connector”](#) and [Section 13.1.3, “Overview of HTTP Connectors”](#).

Prerequisites

- » Either Enterprise Web Server HTTPD or Apache HTTPD needs to be installed. A standalone HTTPD is provided as a separate download in the Red Hat Customer Portal at <https://access.redhat.com>, in

the JBoss Enterprise Application Platform 6 download area. Refer to [Section 13.2.9, “Install the Apache HTTPD included with JBoss Enterprise Application Platform 6”](#) for information about this HTTPD if you wish to use it.

- ▶ The **mod_proxy** modules need to be installed. Apache HTTPD typically comes with the **mod_proxy** modules already included. This is the case on Red Hat Enterprise Linux, the HTTPD that comes with JBoss Enterprise Web Server, and the Apache HTTPD that comes with Microsoft Windows.
- ▶ You need **root** or administrator privileges to modify the HTTPD configuration.
- ▶ Determine the HTTPD configuration directory. This is the directory containing the **conf/** and **modules/** directories for Apache HTTPD. This will be referred to as **HTTPD_CONF** for the remainder of this task. Typical values include the following:
 - **/etc/httpd/**
 - **EWS_HOME/httpd/**, starting from where Enterprise Web Server is installed
- ▶ JBoss Enterprise Application Platform must be configured with the HTTP or HTTPS web connector. This is part of the Web subsystem configuration. Refer to [Section 13.2.2, “Configure the Web Subsystem”](#) for information about configuring the Web subsystem.

1. Enable the mod_proxy modules in the HTTPD

Look for the following lines in your **HTTPD_CONF/conf/httpd.conf** file. If they are not present, add them to the bottom. If they are present but the lines begin with a comment (#) character, remove the character. Save the file afterward. Usually, the modules are already present and enabled.

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_http_module modules/mod_proxy_http.so
# Uncomment these to proxy FTP or HTTPS
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
```

2. Add a non-load-balancing proxy.

Add the following configuration to your **HTTPD_CONF/conf/httpd.conf** file, directly beneath any other **<VirtualHost>** directives you may have. Replace the values with ones appropriate to your set-up.

This example uses a virtual host. See the next step to use the default HTTPD configuration.

```
<VirtualHost * :80>
# Your domain name
ServerName Domain_NAME_HERE

ProxyPreserveHost On

# The IP and port of the JBoss Enterprise Application Platform
# These represent the default values, if your HTTPD is on the same host
# as your JBoss Enterprise Application Platform 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>
```

After making your changes, save the file.

3. Add a load-balancing proxy.

To use **mod_proxy** as a load balancer, and send work to multiple JBoss Enterprise Application Platform servers, add the following configuration to your **HTTPD_CONF/conf/httpd.conf** file.

```
<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://127.0.0.1:8080 route=node1
BalancerMember http://127.0.0.1:8180 route=node2
</Proxy>

# Use the balancer as a single proxy, as in the <VirtualHost> example above.
ProxyPass /::/mycluster
ProxyPassReverse / http://127.0.0.1:8080/
ProxyPassReverse / http://127.0.0.1:8180/
# Only proxy a specific application
# ProxyPass /MyApp::/mycluster
# ProxyPassReverse /MyApp http://host3:8280/MyApp

Starting the httpd ends with
Syntax error on line 1023 of /etc/httpd/conf/httpd.conf:
ProxyPass|ProxyPassMatch needs a path when not defined in a location

Ends with
Syntax error on line 1023 of /etc/httpd/conf/httpd.conf:
ProxyPass|ProxyPassMatch needs a path when not defined in a location

Start of Httpd without errors is expected.
```

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. Refer to the Apache mod_cluster documentation for more details.

4. Enable sticky sessions.

Sticky sessions mean that if a client request originally goes to a specific JBoss Enterprise Application Platform node, all future requests will be sent to the same node, unless the node becomes unavailable. This is almost always the correct behavior.

To enable sticky sessions for **mod_proxy**, add the **stickysession** parameter to the **ProxyPass** statement. This example also shows some other parameters which you can use. Refer to Apache's **mod_proxy** documentation at http://httpd.apache.org/docs/2.2/mod/mod_proxy.html for more information on them.

```
ProxyPass /MyApp balancer://mycluster stickysession=JSESSIONID
lbmethod=bytraffic nofailover=Off
```

5. Restart the HTTPD.

Restart the HTTPD server for your changes to take effect.

Result

Your HTTPD is configured to use **mod_proxy** to send client requests to JBoss Enterprise Application Platform servers or clusters, either in a standard or load-balancing configuration. To configure the JBoss Enterprise Application Platform to respond to these requests, refer to [Section 13.2.11. "Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD"](#).

[Report a bug](#)

13.6. Microsoft ISAPI

13.6.1. About the Internet Server API (ISAPI) HTTP Connector

Internet Server API (ISAPI) is the HTTP connector for Microsoft's Internet Information Services (IIS) web server. You can use the JBoss Enterprise Application Platform as a worker node within an IIS cluster.

To configure the JBoss Enterprise Application Platform to participate in an IIS cluster, refer to [Section 13.6.2. "Configure Microsoft IIS to Use the ISAPI Redirector"](#). For more information about ISAPI, refer to [http://msdn.microsoft.com/en-us/library/ms524911\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms524911(v=VS.90).aspx).

[Report a bug](#)

13.6.2. Configure Microsoft IIS to Use the ISAPI Redirector

Prerequisites

- ▶ ISAPI runs on Microsoft Windows Server 2003 and newer. Make sure you are using a supported operating system and install the IIS server.
 - ▶ Download the JBoss Native Components package for Microsoft Windows, from the Customer Support Portal at <https://access.redhat.com>. Navigate to **Downloads**, then **JBoss Middleware**, then **Application Platform**. Choose either **i386** or **x86_64**. Unzip the file, which includes the ISAPI redirect DLL in the **sbin/** directory.
- Extract the Native Components zip file and copy the contents of the **sbin/** directory to a location on your server. The rest of this task assumes that you used **C:\Connectors**.

Depending on your version of IIS, choose one of the following procedures:

- ▶ [Procedure 13.11, “Configure the IIS Redirector Using the IIS Manager \(IIS 7\)”](#)
- ▶ [Procedure 13.12, “Configure the IIS Redirector Using the IIS Manager \(IIS 6\)”](#)

Procedure 13.11. 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**
 - a. Right-click **Default Web Site**, and click **Add Virtual Directory**. The **Add Virtual Directory** window opens.
 - b. Specify the following values to add a virtual directory:
 - ▶ **Alias:** **jboss**
 - ▶ **Physical Path:** **C:\connectors**
 - c. Click **OK** to save the values and close the **Add Virtual Directory** window.
8. **Define a JBoss Native ISAPI Redirect Filter**
 - a. In the tree view pane, expand **Sites** → **Default Web Site**.
 - b. Double-click **ISAPI Filters**. The **ISAPI Filters Features** view appears.
 - c. In the **Actions** pane, click **Add**. The **Add ISAPI Filter** window appears.
 - d. Specify the following values in the **Add ISAPI Filter** window:
 - ▶ **Filter name:** **jboss**
 - ▶ **Executable:** **C:\connectors\isapi_redirect.dll**
 - e. Click **OK** to save the values and close the **Add ISAPI Filters** window.
9. **Enable the ISAPI-dll handler**
 - a. Double-click the **IIS 7** item in the tree view pane. The **IIS 7 Home Features View** opens.
 - b. Double-click **Handler Mappings**. The **Handler Mappings Features View** appears.
 - c. In the **Group by** combo box, select **State**. The **Handler Mappings** are displayed in **Enabled and Disabled Groups**.
 - d. Find **ISAPI-dll**. If it is in the **Disabled** group, right-click it and select **Edit Feature Permissions**.
 - e. Enable the following permissions:
 - ▶ **Read**
 - ▶ **Script**
 - ▶ **Execute**
 - f. Click **OK** to save the values, and close the **Edit Feature Permissions** window.

Procedure 13.12. Configure the IIS Redirector Using the IIS Manager (IIS 6)

1. Open the IIS manager by clicking **Start → Run**, and typing **inetmgr**.
2. In the tree view at the left, expand **Websites** and right-click the website you wish to configure. The rest of this task assumes you are configuring the **Default** website. Click **Properties**.
3. Click the **ISAPI Filters** tab.
4. Click the **Add** button. Name your filter **jboss**, and specify **C:\connectors\isapi_redirect.dll** as the executable.
5. Click OK, and close all dialogs. Leave the IIS Manager open.
6. **Define the ISAPI virtual directory.**
 - a. Right-click the Default website again. Select **NewAdd Virtual Directory**.
 - b. Specify **jboss** as the alias, and **C:\connectors\jboss-ep-6.0\native\sbin** as the physical path.
 - c. Click **OK** to save the values and close the **Add Virtual Directory** window.
 - d. In the tree view, expand **Web Sites/Default Web Site**.
 - e. Right-click the **jboss** virtual directory, and click **Properties**.
 - f. Click the **Virtual Directory** tab, and make the following changes.

Execute permission

 - ▶ Scripts and Executables

Read Access

 - ▶ Activate Read Access by checking the box.
 - g. Click **OK** to save the changes and close the **JBoss Properties** window.
7. **Define ISAPI web service extensions**
 - a. Click **Web Service Extensions**. In the **Tasks** group, select **Add a new Web service extension**. The **New Web Service Extension** window opens.
 - b. Add the following values:
 - ▶ Extension name: **jboss**
 - ▶ Required files: **C:\connectors\isapi_redirect.dll**
 - ▶ Extension Status: **allowed**
 - c. Click **OK** to save the changes and close the **New Web Service Extension** window.
 - d. Confirm that the **jboss** Web Service Extension displays in the list.

Result

Microsoft IIS is now configured to use the ISAPI Redirector. Next, [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#), then [Section 13.6.3, “Configure the ISAPI Redirector to Send Client Requests to the JBoss Enterprise Application Platform”](#) or [Section 13.6.4, “Configure ISAPI to Balance Client Requests Across Multiple JBoss Enterprise Application Platform Servers”](#).

[Report a bug](#)

13.6.3. Configure the ISAPI Redirector to Send Client Requests to the JBoss Enterprise Application Platform

Overview

This task configures a group of JBoss Enterprise Application Platform servers to accept requests from the ISAPI redirector. It does not include configuration for load-balancing or high-availability failover. If you need these capabilities, refer to [Section 13.6.4, “Configure ISAPI to Balance Client Requests Across Multiple JBoss Enterprise Application Platform Servers”](#).

This configuration is done on the IIS server, and assumes that the JBoss Enterprise Application Platform is already configured, as per [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#).

Prerequisites

- ▶ You need full administrator access to the IIS server
- ▶ [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#)
- ▶ [Section 13.6.2, “Configure Microsoft IIS to Use the ISAPI Redirector”](#)

Procedure 13.13. Task

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. Substitute the value **JBOSS_NATIVE_HOME** with the actual location where you installed the JBoss Native components when you performed the task [Section 13.6.2, "Configure Microsoft IIS to Use the ISAPI Redirector"](#).

```
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=JBOSS_NATIVE_HOME/sbin/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.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. See [Step 5](#) for more information.

3. Create the `uriworkermap.properties` file

The `uriworkermap.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 `uriworkermap.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. The following example file shows the syntax of the file. 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 Enterprise Application Platform 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 Enterprise Application Platform 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 the IIS server.

Follow the appropriate procedure for restarting your IIS server, depending on its version.

A. IIS 6

```
C:\> net stop iisadmin /Y
C:\> net start w3svc
```

B. IIS 7

```
C:\> net stop was /Y
C:\> net start w3svc
```

Result

The IIS server is configured to send client requests to the specific JBoss Enterprise Application Platform servers you have configured, on an application-specific basis.

[Report a bug](#)

13.6.4. Configure ISAPI to Balance Client Requests Across Multiple JBoss Enterprise Application Platform Servers

Overview

This configuration balances client requests across the JBoss Enterprise Application Platform servers you specify. If you prefer to send client requests to specific JBoss Enterprise Application Platform servers on a per-deployment basis, refer to [Section 13.6.3, “Configure the ISAPI Redirector to Send Client Requests to the JBoss Enterprise Application Platform”](#) instead.

This configuration is done on the IIS server, and assumes that the JBoss Enterprise Application Platform is already configured, as per [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#).

Prerequisites

- » Full administrator access on the IIS server.
- » [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#)
- » [Section 13.6.2, “Configure Microsoft IIS to Use the ISAPI Redirector”](#)

Procedure 13.14. Task

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. Substitute the value **JBOSS_NATIVE_HOME** with the actual location where you installed the JBoss Native components when you performed the task [Section 13.6.2, "Configure Microsoft IIS to Use the ISAPI Redirector"](#).

```
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=JBOSS_NATIVE_HOME/sbin/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.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. See [Step 5](#) for more information.

3. Create the **uriworkermap.properties** file.

The **uriworkermap.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 [Step 4](#).

Place your **uriworkermap.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. The following example file shows the syntax of the file. The load balancer is configured near the end of the file, to comprise workers **worker01** and **worker02**. The **workers.properties** file follows the syntax of the same file used for Apache mod_jk configuration. For more information about the syntax of the **workers.properties** file, refer to [Section 13.4.4, "Configuration Reference for Apache Mod_jk Workers"](#).

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/

```

6. Restart the IIS server.

Follow the appropriate procedure for restarting your IIS server, depending on its version.

A. IIS 6

```

C:\> net stop iisadmin /Y
C:\> net start w3svc

```

B. IIS 7

```

C:\> net stop was /Y
C:\> net start w3svc

```

Result

The IIS server is configured to send client requests to the JBoss Enterprise Application Platform servers referenced in the `workers.properties` file, balancing the load equally across the servers.

[Report a bug](#)

13.7. Oracle NSAPI

13.7.1. About the Netscape Server API (NSAPI) HTTP Connector

The *Netscape Server API (NSAPI)* is the HTTP connector that allows the JBoss Enterprise Application Platform to participate as a node in Oracle iPlanet Web Server (formerly Netscape Web Server). To configure this connector, refer to [Section 13.7.4, “Configure NSAPI as a Load-balancing Cluster”](#).

[Report a bug](#)

13.7.2. Configure the NSAPI Connector on Oracle Solaris

Overview

The NSAPI connector is a module that runs within Oracle iPlanet Web Server.

Prerequisites

- ▶ Your server is running Oracle Solaris 9 or greater, on either a 32-bit or 64-bit architecture.
- ▶ Oracle iPlanet Web Server 6.1 SP 12 or 7.0 U8 is installed and configured, aside from the NSAPI connector.
- ▶ The JBoss Enterprise Application Platform is installed and configured on each server which will serve as a worker node. Refer to [Section 13.2.11, “Configure the JBoss Enterprise Application Platform to Accept Requests From an External HTTPD”](#).
- ▶ The JBoss Native Components ZIP package is downloaded from the Customer Service Portal at <https://access.redhat.com>.

Procedure 13.15. Task

1. Extract the JBoss Native Components package.

The rest of this procedure assumes that the Native Components package is extracted to a directory called **connectors/** in **/opt/oracle/webserver7/config/**. For the rest of this procedure, this directory will be referred to as **IPLANET_CONFIG**. If your Oracle iPlanet configuration directory is different, or you are running Oracle iPlanet Web Server 6, modify the procedure accordingly.

2. 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.

3. 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 worker nodes and their properties.

```
Init fn="load-modules" funcs="jk_init,jk_service"
shlib="IPLANET_CONFIG/connectors/lib/nsapi_redirector.so"
shlib_flags="(global|now)"
Init fn="jk_init" worker_file="IPLANET_CONFIG/connectors/workers.properties"
log_level="debug" log_file="IPLANET_CONFIG/config/connectors/nsapi.log"
shm_file="IPLANET_CONFIG/conf/connectors/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.

4. 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.

- ▶ [Section 13.7.3, “Configure NSAPI as a Basic HTTP Connector”](#)
- ▶ [Section 13.7.4, “Configure NSAPI as a Load-balancing Cluster”](#)

[Report a bug](#)

13.7.3. Configure NSAPI as a Basic HTTP Connector

Overview

This task configures the NSAPI connector to redirect client requests to JBoss Enterprise Application Platform servers with no load-balancing or fail-over. The redirection is done on a per-deployment (and hence per-URL) basis. For a load-balancing configuration, refer to [Section 13.7.4, “Configure NSAPI as a Load-balancing Cluster”](#) instead.

Prerequisites

- ▶ You must complete [Section 13.7.2, “Configure the NSAPI Connector on Oracle Solaris”](#) before continuing with the current task.

Procedure 13.16. Task

1. Define the URL paths to redirect to the JBoss Enterprise Application Platform servers.

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 **jkn sapi** refers to the HTTP connector which will be defined in the next step. The example shows the use of wild-cards for pattern matching.

```
<Object name="default">
  [...]
  NameTrans fn="assign-name" from="/status" name="jkn sapi"
  NameTrans fn="assign-name" from="/images/*" name="jkn sapi"
  NameTrans fn="assign-name" from="/css/*" name="jkn sapi"
  NameTrans fn="assign-name" from="/nc/*" name="jkn sapi"
  NameTrans fn="assign-name" from="/jmx-console/*" name="jkn sapi"
</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="jkn sapi">
  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 **jkn sapi** 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. For information about which options are available, refer to [Section 13.4.4, “Configuration Reference for Apache Mod_jk Workers”](#).

Save and exit the file.

4. Restart the iPlanet Web Server.

Choose one of the following procedures, depending on whether you run iPlanet Web Server 6.1 or 7.0.

A. iPlanet Web Server 6.1

```
IPLANET_CONFIG/.../stop
IPLANET_CONFIG/.../start
```

B. iPlanet Web Server 7.0

```
IPLANET_CONFIG/.../bin/stopserv
IPLANET_CONFIG/.../bin/startserv
```

Result

iPlanet Web Server now sends client requests to the URLs you have configured to deployments on the JBoss Enterprise Application Platform.

[Report a bug](#)

13.7.4. Configure NSAPI as a Load-balancing Cluster

Overview

This task configures the NSAPI connector to redirect client requests to JBoss Enterprise Application Platform servers in a load-balancing configuration. To use NSAPI as a simple HTTP connector with no load-balancing, refer to [Section 13.7.3, “Configure NSAPI as a Basic HTTP Connector”](#) instead.

Prerequisites

- ▶ You must complete [Section 13.7.2, “Configure the NSAPI Connector on Oracle Solaris”](#) before continuing with the current task.

Procedure 13.17. Task

1. Define the URL paths to redirect to the JBoss Enterprise Application Platform servers.

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 **jkn sapi** refers to the HTTP connector which will be defined in the next step. The example shows the use of wild-cards for pattern matching.

```
<Object name="default">
  [...]
  NameTrans fn="assign-name" from="/status" name="jkn sapi"
  NameTrans fn="assign-name" from="/images(/*)" name="jkn sapi"
  NameTrans fn="assign-name" from="/css(/*)" name="jkn sapi"
  NameTrans fn="assign-name" from="/nc(/*)" name="jkn sapi"
  NameTrans fn="assign-name" from="/jmx-console(/*)" name="jkn sapi"
  NameTrans fn="assign-name" from="/jkmanager/*" name="jkn sapi"
</Object>
```

2. 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 **jknsapi** object defines the worker nodes used to serve each path that was mapped to the **name="jknsapi"** 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/conf/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 Enterprise Application Platform 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 Enterprise Application Platform 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. For information about which options are available, refer to [Section 13.4.4, “Configuration Reference for Apache Mod_jk Workers”](#).

Save and exit the file.

4. Restart the iPlanet Web Server.

Choose one of the following procedures, depending on whether you run iPlanet Web Server 6.1 or 7.0.

A. iPlanet Web Server 6.1

```
IPLANET_CONFIG/.../stop
IPLANET_CONFIG/.../start
```

B. iPlanet Web Server 7.0

```
IPLANET_CONFIG/.../bin/stopserv
IPLANET_CONFIG/.../bin/startserv
```

Result

The iPlanet Web Server redirects the URL patterns you have configured to your JBoss Enterprise Application Platform servers in a load-balancing configuration.

[Report a bug](#)

Chapter 14. Messaging

14.1. HornetQ

14.1.1. HornetQ

HornetQ is a multi-protocol, asynchronous messaging system developed by Red Hat. HornetQ provides high availability (HA) with automatic client failover to guarantee message reliability in the event of a server failure. HornetQ also supports flexible clustering solutions with load-balanced messages.

[Report a bug](#)

14.1.2. About Java Messaging Service (JMS)

Messaging systems allow you to loosely couple heterogeneous systems together with added reliability. Java Messaging Service (JMS) providers use a system of transactions, to commit or roll back changes atomically. Unlike systems based on a Remote Procedure Call (RPC) pattern, messaging systems primarily use an asynchronous message passing pattern with no tight relationship between requests and responses. Most messaging systems also support a request-response mode but this is not a primary feature of messaging systems.

Messaging systems decouple the senders of messages from the consumers of messages. The senders and consumers of messages are completely independent and know nothing of each other. This allows you to create flexible, loosely coupled systems. Often, large enterprises use a messaging system to implement a message bus which loosely couples heterogeneous systems together. Message buses often form the core of an Enterprise Service Bus (ESB). Using a message bus to decouple disparate systems can allow the system to grow and adapt more easily. It also allows more flexibility to add new systems or retire old ones since they don't have brittle dependencies on each other.

[Report a bug](#)

14.1.3. Supported Messaging Styles

HornetQ supports the following messaging styles:

Message Queue pattern

The Message Queue pattern involves sending a message to a queue. Once in the queue, the message is usually made persistent to guarantee delivery. Once the message has moved through the queue, the messaging system delivers it to a message consumer. The message consumer acknowledges the delivery of the message once it is processed.

When used with point-to-point messaging, the Message Queue pattern allows multiple consumers for a queue, but each message can only be received by a single consumer.

Publish-Subscribe pattern

The Publish-Subscribe pattern allows multiple senders to send messages to a single entity on the server. This entity is often known as a "topic". Each topic can be attended by multiple consumers, known as "subscriptions".

Each subscription receives a copy of every message sent to the topic. This differs from the Message Queue pattern, where each message is only consumed by a single consumer.

Subscriptions that are durable retain copies of each message sent to the topic until the subscriber consumes them. These copies are retained even in the event of a server restart. Non-durable subscriptions last only as long as the connection that created them.

[Report a bug](#)

14.1.4. About Acceptors and Connectors

HornetQ uses the concept of connectors and acceptors as a key part of the messaging system.

Acceptors and Connectors

Acceptor

An acceptor defines which types of connections are accepted by the HornetQ server.

Connector

A connector defines how to connect to a HornetQ server, and is used by the HornetQ client.

There are two types of connectors and acceptors, relating to the whether the matched connector and acceptor pair occur within same JVM or not.

Invm and Netty

Invm

Invm is short for Intra Virtual Machine. It can be used when both the client and the server are running in the same JVM.

Netty

The name of a JBoss project. It must be used when the client and server are running in different JVMs.

A HornetQ client must use a connector that is compatible with one of the server's acceptors. Only an Invm connector can connect to an Invm acceptor, and only a netty connector can connect to a netty acceptor. The connectors and acceptors are both configured on the server in a **standalone.xml** and **domain.xml**. You can use either the Management Console or the Management CLI to define them.

Example 14.1. Example of the Default Acceptor and Connector Configuration

```
<connectors>
    <netty-connector name="netty" socket-binding="messaging"/>
    <netty-connector name="netty-throughput" socket-binding="messaging-
throughput">
        <param key="batch-delay" value="50"/>
    </netty-connector>
    <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
<acceptors>
    <netty-acceptor name="netty" socket-binding="messaging"/>
    <netty-acceptor name="netty-throughput" socket-binding="messaging-
throughput">
        <param key="batch-delay" value="50"/>
        <param key="direct-deliver" value="false"/>
    </netty-acceptor>
    <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>
```

The example configuration also shows how the JBoss Enterprise Application Platform 6 implementation of HornetQ uses socket bindings in the acceptor and connector configuration. This differs from the standalone version of HornetQ, which requires you to declare the specific hosts and ports.

[Report a bug](#)

14.1.5. About Bridges

The function of a bridge is to consume messages from a source queue, and forward them to a target address, typically on a different HornetQ server. Bridges cope with unreliable connections, automatically reconnecting when the connections become available again. HornetQ bridges can be configured with filter expressions to only forward certain messages.

[Report a bug](#)

14.1.6. Work with Large Messages

HornetQ supports the use of large messages even when either the client or server has limited amounts of memory. Large messages can be streamed as they are, or compressed further for more efficient transferral.

[Report a bug](#)

14.1.7. Configure High-availability (HA) Failover

High-availability failover is available with either automatic client failover or application-level failover.



Important

HornetQ HA only supports shared stores on GFS2 on SAN.

[Report a bug](#)

14.1.8. Embed HornetQ in Applications

To implement messaging functionality internally within an application, you can directly instantiate HornetQ clients and servers in the application code. This is called *embedding* HornetQ.

1. Instantiate the configuration object

Instantiate the configuration object either from a configuration file, or by setting configuration parameters programmatically.

A. Create the configuration object from a file

Use the `FileConfigurationImpl` class to set a configuration object based on a file.

```
import org.hornetq.core.config.Configuration;
import org.hornetq.core.config.impl.FileConfiguration;
...
Configuration config = new FileConfiguration();
config.setConfigurationUrl(<replaceable>file-url</replaceable>);
config.start();
```

B. Create the configuration object programmatically

a. Use the `ConfigurationImpl` class to create a configuration object.

```
import org.hornetq.core.config.Configuration;
import org.hornetq.core.config.impl.FileConfiguration;
...
Configuration config = new ConfigurationImpl();
```

b. Set any configuration parameters programmatically, such as acceptors. For example:

```
HashSet<TransportConfiguration> transports = new
HashSet<TransportConfiguration>();

transports.add(new
TransportConfiguration(NettyAcceptorFactory.class.getName()));
transports.add(new
TransportConfiguration(InVMAcceptorFactory.class.getName()));

config.setAcceptorConfigurations(transports);
```

2. Instantiate and start the server

Use the `org.hornetq.api.core.server.HornetQ` static methods to create and start a server based on the configuration object.

```
import org.hornetq.api.core.server.HornetQ;
import org.hornetq.core.server.HornetQServer;
...
HornetQServer server = HornetQ.newHornetQServer(config);
server.start();
```

Result:

An embedded HornetQ instance is instantiated for internal messaging. To connect clients to the embedded HornetQ, create factories as normal.

[Report a bug](#)

14.1.9. Configure the JMS Server

To configure the JMS Server for HornetQ, edit the server configuration file. The server configuration is

contained in the `EAP_HOME/domain/configuration/domain.xml` file for domain servers, or in the `EAP_HOME/standalone/configuration/standalone.xml` file for standalone servers.

The `<subsystem xmlns="urn:jboss:domain:messaging:1.2">` element contains all JMS configuration. Add any JMS **ConnectionFactory**, **Queue**, or **Topic** instances required for the JNDI.

1. Enable the JMS subsystem in the JBoss Enterprise Application Platform.

In the `<extensions>` element, verify that the following line is present and is not commented out:

```
<extension module="org.jboss.as.messaging"/>
```

2. Add the basic JMS subsystem.

If the Messaging subsystem is not present in your configuration file, add it.

- Look for the `<profile>` which corresponds to the profile you use, and locate its `<subsystems>` tag.
- Add a new line just beneath the `<subsystems>` tag. Paste the following into it:

```
<subsystem xmlns="urn:jboss:domain:messaging:1.2">
</subsystem>
```

All further configuration will be added to the empty line above.

3. Add basic configuration for JMS.

```
<journal-file-size>102400</journal-file-size>
<journal-min-files>2</journal-min-files>
<journal-type>NIO</journal-type>
<!-- disable messaging persistence -->
<persistence-enabled>false</persistence-enabled>
```

Customize the values above to meet your needs.

4. Add connection factory instances to HornetQ

The client uses a JMS **ConnectionFactory** object to make connections to the server. To add a JMS connection factory object to HornetQ, include a single `<jms-connection-factories>` tag and `<connection-factory>` element for each connection factory as follows:

```
<subsystem xmlns="urn:jboss:domain:messaging:1.2">
...
<jms-connection-factories>
  <connection-factory name="myConnectionFactory">
    <connectors>
      <connector-ref connector-name="netty"/>
    </connectors>
    <entries>
      <entry name="/ConnectionFactory"/>
    </entries>
  </connection-factory>
</jms-connection-factories>
...
</subsystem>
```

5. Configure the netty connector

This JMS connection factory uses a **netty** connector. This is a reference to a connector object deployed in the server configuration file. The connector object defines the transport and parameters used to actually connect to the server.

To configure the **netty** connector, include the following settings:

```
<subsystem xmlns="urn:jboss:domain:messaging:1.2">
  ...
  <connectors>
    <netty-connector name="netty" socket-binding="messaging"/>
    <netty-connector name="netty-throughput" socket-binding="messaging-
throughput">
      <param key="batch-delay" value="50"/>
    </netty-connector>
    <in-vm-connector name="in-vm" server-id="0"/>
  </connectors>
  ...
</subsystem>
```

The connector references the **messaging** and **messaging-throughput** socket bindings. The **messaging** socket binding uses port 5445, and the **messaging-throughput** socket binding uses port 5455. Ensure the following socket bindings are present in the **<socket-binding-groups>** element:

```
<socket-binding-groups>
  ...
  <socket-binding-group ... >
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-throughput" port="5455"/>
  ...
</socket-binding-group>
...
</socket-binding-groups>
```

6. Add queue instances to HornetQ

The client uses a JMS **Queue** object to stage sent messages for delivery to the server. To add a JMS queue object to HornetQ, include a **<jms-queue>** element as follows:

```
<subsystem xmlns="urn:jboss:domain:messaging:1.2">
  ...
  <jms-destinations>
    <jms-queue name="myQueue">
      <entry name="/queue/myQueue"/>
    </jms-queue>
  </jms-destinations>
  ...
```

7. Optional: Add topic instances to HornetQ

The client uses a JMS **Topic** object to manage messages for multiple subscribers. To add a JMS topic object, include a **<topic>** element as follows:

```
<subsystem xmlns="urn:jboss:domain:messaging:1.2">
  ...
  <jms-topic name="myTopic">
    <entry name="/topic/myTopic"/>
  </jms-topic>
  ...
```

8. Perform additional configuration

If you need additional settings, review the DTD in **EAP_HOME/docs/schema/jboss-messaging_1_2.xsd**.

[Report a bug](#)

14.1.10. About Java Naming and Directory Interface (JNDI)

The *Java Naming and Directory Interface (JNDI)* is a standard Java API for naming and directory services. It allows Java-based technologies to discover and organize named components in a distributed computing environment.

[Report a bug](#)

14.1.11. Configure JNDI for HornetQ

Prerequisites

- ▶ [Section 2.6.2, "Start JBoss Enterprise Application Platform 6 as a Standalone Server"](#)
- ▶ [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 14.1. Task

When the JBoss Enterprise Application Platform runs as a managed domain, a JNDI server is provided and does not need configuration.

Follow this procedure to configure the **JNDIServer** bean for HornetQ when the JBoss Enterprise Application Server is running as a standalone server.

1. Set the JNDI properties on the client side

The JNDI properties tell the JNDI client where to locate the JNDI server. The properties can be specified in a file named **jndi.properties** on the client classpath, or declared directly when creating the initial JNDI context.

Set the JNDI properties to the following settings:

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.provider.url=jnp://hostname:1099
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
```

▶ **hostname** is the hostname or IP address of the JNDI server.

2. Configure the JNDI server ports

a. Navigate to the Socket Binding Groups panel in the Management Console

Select the **General Configuration** → **Socket Binding Groups** option from the menu on the left of the console.

The **Socket Binding Groups** panel for the selected server appears.

b. Edit the JNDI Socket Binding

Select **jndi** from the **Socket Binding Declarations** table, then select the **Edit** button in the **Socket Binding** section below.

Configure the following settings: and **Multicast Port** settings. Select the **Save** button when done.

▶ **Port:1099**

▶ **Multicast Port:1098**

Result

The JNDI server is now configured for HornetQ.

[Report a bug](#)

14.1.12. Configure JMS Address Settings

The JMS subsystem has several configurable options which control aspects of how and when a message is delivered, how many attempts should be made, and when the message expires. These configuration options all exist within the **<address-settings>** configuration element.

A common feature of address configurations is the syntax for matching multiple addresses, also known as wild cards.

Wildcard Syntax

Address wildcards can be used to match multiple similar addresses with a single statement, similar to how many systems use the asterisk (*) character to match multiple files or strings with a single search. The following characters have special significance in a wildcard statement.

Table 14.1. JMS Wildcard Syntax

Character	Description
. (a single period)	Denotes the space between words in a wildcard expression.
# (a pound or hash symbol)	Matches any sequence of zero or more words.
* (an asterisk)	Matches a single word.

Table 14.2. JMS Wildcard Examples

Example	Description
news.europe.#	Matches news.europe , news.europe.sport , news.europe.politic , but not news.usa or europe .
news.	Matches news.europe but not news.europe.sport .
news.*.sport	Matches news.europe.sport and news.usa.sport , but not news.europe.politics .

Example 14.2. Default Address Setting Configuration

The values in this example are used to illustrate the rest of this topic.

```
<address-settings>
    <!--default for catch all-->
    <address-setting match="#">
        <dead-letter-address>jms.queue.DLQ</dead-letter-address>
        <expiry-address>jms.queue.ExpiryQueue</expiry-address>
        <redelivery-delay>0</redelivery-delay>
        <max-size-bytes>10485760</max-size-bytes>
        <address-full-policy>BLOCK</address-full-policy>
        <message-counter-history-day-limit>10</message-counter-history-day-
limit>
    </address-setting>
</address-settings>
```

Table 14.3. Description of JMS Address Settings

Element	Description	Default Value	Type
address-full-policy	Determines what happens when an address where max-size-bytes is specified becomes full.	PAGE	STRING
dead-letter-address	If a dead letter address is specified, messages are moved to the dead letter address if max-delivery-attempts delivery attempts have failed. Otherwise, these undelivered messages are discarded. Wildcards are allowed.	jms.queue.DLQ	STRING
expiry-address	If the expiry address is present, expired messages are sent to the address or addresses matched by it, instead of being discarded. Wildcards are allowed.	jms.queue.ExpiryQueue	STRING
last-value-queue	Defines whether a queue only uses last values or not.	false	BOOLEAN
max-delivery-attempts	The maximum number of times to attempt to re-deliver a message before it is sent to dead-letter-address or discarded.	10	INT
max-delivery-bytes	The maximum bytes size.	10485760L	LONG
message-counter-history-day-limit	Day limit for the message counter history.	10	INT
page-max-cache-size	The number of page files to keep in memory to optimize IO during paging navigation.	5	INT
page-size-bytes	The paging size.	5	INT
redelivery-delay	Time to delay between re-delivery attempts of messages, expressed in milliseconds. If set to 0, re-delivery attempts occur indefinitely.	0L	LONG
redistribution-delay	Defines how long to wait when the last consumer is closed on a queue before redistributing any messages.	-1L	LONG
send-to-dla-on-no-route	A parameter for an address that sets the condition of a message not ruoted to any queues to instead be sent the to the dead letter address (DLA)	false	BOOLEAN

indicated for that address.

Configure Address Setting and Pattern Attributes

Choose either the Management CLI or the Management Console to configure your pattern attributes as required.

A. Configure the Address Settings Using the Management CLI

Use the Management CLI to configure address settings.

1. Add a New Pattern

Use the **add** operation to create a new address setting if required. You can run this command from the root of the Management CLI session, which in the following examples creates a new pattern titled **patternname**, with a **max-delivery-attempts** attribute declared as **5**. The examples for both Standalone Server and a Managed Domain editing on the **full** profile are shown.

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
```

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
```

2. Edit Pattern Attributes

Use the **write** operation to write a new value to an attribute. You can use tab completion to help complete the command string as you type, as well as to expose the available attributes. The following example updates the **max-delivery-attempts** value to **10**.

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:write-attribute(name=max-delivery-attempts,value=10)
```

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:write-attribute(name=max-delivery-attempts,value=10)
```

3. Confirm Pattern Attributes

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:read-resource
```

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:read-resource
```

B. Configure the Address Settings Using the Management Console

Use the Management Console to configure address settings.

1. Log into the Management Console.

Log into the Management Console of your Managed Domain or Standalone Server.

2. If you use a Managed Domain, choose the correct profile.

Select the **Profiles** tab at the top right, and then select the correct profile from the **Profile** menu at the top left of the next screen. Only the **full** and **full-ha** profiles have the **messaging** subsystem enabled.

3. Select the Messaging item from the navigation menu.

Expand the **Messaging** menu item from the navigation menu, and click **Destinations**.

4. View the JMS Provider.

A list of JMS Providers is shown. In the default configuration, only one provider, called **default**, is shown. Click the **View** link to view the detailed settings for this provider.

5. View the Address Settings.

Click the **Addressing** tab. Either add a new pattern by clicking the **Add** button, or edit an existing one by clicking its name and clicking the **Edit** button.

6. Configure the options.

If you are adding a new pattern, the **Pattern** field refers to the **match** parameter of the **address-setting** element. You can also edit the **Dead Letter Address**, **Expiry Address**, **Redelivery Delay**, and **Max Delivery Attempts**. Other options need to be configured using the Management CLI.

[Report a bug](#)

14.1.13. Reference for HornetQ Configuration Attributes

The JBoss Enterprise Application Platform 6 implementation of HornetQ exposes the following attributes for configuration. You can use the Management CLI in particular to exposure the configurable or viewable attributes with the **read-resource** operation.

Example 14.3. Example

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default:read-resource
```

Table 14.4. TableTitle

Attribute	Example Value	Type
<code>allow-failback</code>	true	BOOLEAN
<code>async-connection-execution-enabled</code>	true	BOOLEAN
<code>backup</code>	false	BOOLEAN
<code>cluster-password</code>	<i>somethingsecure</i>	STRING
<code>cluster-user</code>	HORNETQ.CLUSTER.A DMIN.USER	STRING
<code>clustered</code>	false	BOOLEAN
<code>connection-ttl-override</code>	-1	LONG
<code>create-bindings-dir</code>	true	BOOLEAN
<code>create-journal-dir</code>	true	BOOLEAN
<code>fallback-delay</code>	5000	LONG
<code>failover-on-shutdown</code>	false	BOOLEAN
<code>id-cache-size</code>	2000	INT
<code>jmx-domain</code>	org.hornetq	STRING
<code>jmx-management-enabled</code>	false	BOOLEAN
<code>journal-buffer-size</code>	100	LONG
<code>journal-buffer-timeout</code>	100	LONG
<code>journal-compact-min-files</code>	10	INT
<code>journal-compact-percentage</code>	30	INT
<code>journal-file-size</code>	102400	LONG
<code>journal-max-io</code>	1	INT
<code>journal-min-files</code>	2	INT
<code>journal-sync-non-transactional</code>	true	BOOLEAN
<code>journal-sync-transactional</code>	true	BOOLEAN
<code>journal-type</code>	ASYNCIO	STRING
<code>live-connector-ref</code>	reference	STRING
<code>log-journal-write-rate</code>	false	BOOLEAN
<code>management-address</code>	jms.queue.hornetq.management	STRING
<code>management-notification-address</code>	hornetq.notifications	STRING
<code>memory-measure-interval</code>	-1	LONG
<code>memory-warning-threshold</code>	25	INT
<code>message-counter-enabled</code>	false	BOOLEAN
<code>message-counter-max-day-history</code>	10	INT

message-counter-sample-period	10000	LONG
message-expiry-scan-period	30000	LONG
message-expiry-thread-priority	3	INT
page-max-concurrent-io	5	INT
perf-blast-pages	-1	INT
persist-delivery-count-before-delivery	false	BOOLEAN
persist-id-cache	true	BOOLEAN
persistence-enabled	true	BOOLEAN
remoting-interceptors	undefined	LIST
run-sync-speed-test	false	BOOLEAN
scheduled-thread-pool-max-size	5	INT
security-domain	other	STRING
security-enabled	true	BOOLEAN
security-invalidation-interval	10000	LONG
server-dump-interval	-1	LONG
shared-store	true	BOOLEAN
started	true	BOOLEAN
thread-pool-max-size	30	INT
transaction-timeout	300000	LONG
transaction-timeout-scan-period	1000	LONG
version	2.2.16.Final (HQ_2_2_16_FINAL, 122)	STRING
wild-card-routing-enabled	true	BOOLEAN

[Report a bug](#)

14.1.14. Configure Messaging with HornetQ

The recommended method of configuring messaging in JBoss Enterprise Application Platform 6 is in either the Management Console or Management CLI. You can make persistent changes with either of these management tools without needing to manually edit the `standalone.xml` or `domain.xml` configuration files. It is useful however to familiarise yourself with the messaging components of the default configuration files, where documentation examples using management tools give configuration file snippets for reference.

[Report a bug](#)

14.1.15. Configure Delayed Redelivery

[Introduction](#)

Delayed redelivery is defined in the `<redelivery-delay>` element, which is a child element of the `<address-setting>` configuration element in the Java Messaging Service (JMS) subsystem configuration.

```
<!-- delay redelivery of messages for 5s -->
<address-setting match="jms.queue.exampleQueue">
  <redelivery-delay>5000</redelivery-delay>
</address-setting>
```

If a redelivery delay is specified, the JMS system waits for the duration of this delay before redelivering the messages. If `<redelivery-delay>` is set to `0`, there is no redelivery delay. Address wildcards can be used on the `<address-setting-match>` element to configure the redelivery delay for addresses which match the wildcard.

[Report a bug](#)

14.1.16. Configure Dead Letter Addresses

Introduction

A dead letter address is defined in the `<address-setting>` element of the Java Messaging Service (JMS) subsystem configuration.

```
<!-- undelivered messages in exampleQueue will be sent to the dead letter address
deadLetterQueue after 3 unsuccessful delivery attempts
-->
<address-setting match="jms.queue.exampleQueue">
  <dead-letter-address>jms.queue.deadLetterQueue</dead-letter-address>
  <max-delivery-attempts>3</max-delivery-attempts>
</address-setting>
```

If a `<dead-letter-address>` is not specified, messages are removed after trying to deliver `<max-delivery-attempts>` times. By default, messages delivery is attempted 10 times. Setting `<max-delivery-attempts>` to `-1` allows infinite redelivery attempts. For example, a dead letter can be set globally for a set of matching addresses and you can set `<max-delivery-attempts>` to `-1` for a specific address setting to allow infinite redelivery attempts only for this address. Address wildcards can also be used to configure dead letter settings for a set of addresses.

[Report a bug](#)

14.1.17. Configure Message Expiry Addresses

Introduction

Message expiry addresses are defined in the address-setting configuration of the Java Messaging Service (JMS). For example:

```
<!-- expired messages in exampleQueue will be sent to the expiry address
expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
  <expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

If messages are expired and no expiry address is specified, messages are simply removed from the queue and dropped. Address wildcards can also be used to configure specific ranges of an expiry address for a set of addresses.

Address Wildcards

Address wildcards can be used to match multiple similar addresses with a single statement, similar to how many systems use the asterisk (*) character to match multiple files or strings with a single search. The following characters have special significance in a wildcard statement.

Table 14.5. JMS Wildcard Syntax

Character	Description
. (a single period)	Denotes the space between words in a wildcard expression.
# (a pound or hash symbol)	Matches any sequence of zero or more words.
* (an asterisk)	Matches a single word.

Table 14.6. JMS Wildcard Examples

Example	Description
news.europe.#	Matches news.europe , news.europe.sport , news.europe.politic , but not news.usa or europe .
news.	Matches news.europe but not news.europe.sport .
news.*.sport	Matches news.europe.sport and news.usa.sport , but not news.europe.politics .

[Report a bug](#)

14.1.18. Set Message Expiry

Introduction

Using HornetQ Core API, the expiration time can be set directly on the message. For example:

```
// message will expire in 5000ms from now
message.setExpiration(System.currentTimeMillis() + 5000);
```

JMS MessageProducer

JMS **MessageProducer** includes a **TimeToLive** parameter which controls message expiry for the messages it sends::

```
// messages sent by this producer will be retained for 5s (5000ms) before
// expiration
producer.setTimeToLive(5000);
```

Expired messages which are consumed from an expiry address have the following properties:

▶ **_HQ_ORIG_ADDRESS**

A string property containing the original address of the expired message.

▶ **_HQ_ACTUAL_EXPIRY**

A long property containing the actual expiration time of the expired message.

[Report a bug](#)

Chapter 15. Transaction Subsystem

15.1. Transaction Subsystem Configuration

15.1.1. Transactions Configuration Overview

Introduction

The following procedures show you how to configure the transactions subsystem of the JBoss Enterprise Application Platform.

- ▶ [Section 15.1.3. "Configure Your Datasource to Use JTA Transactions"](#)
- ▶ [Section 15.1.4. "Configure an XA Datasource"](#)
- ▶ [Section 15.1.2. "Configure the Transaction Manager"](#)
- ▶ [Section 15.1.6. "Configure Logging for the Transaction Subsystem"](#)

[Report a bug](#)

15.1.2. Configure the Transaction Manager

You can configure the Transaction Manager (TM) using the web-based Management Console or the command-line Management CLI. For each command or option given, the assumption is made that you are running JBoss Enterprise Application Platform 6 as a Managed Domain. If you use a Standalone Server or you want to modify a different profile than **default**, you may need to modify the steps and commands in the following ways.

Notes about the Example Commands

- ▶ For the Management Console, the **default** profile is the one which is selected when you first log into the console. If you need to modify the Transaction Manager's configuration in a different profile, select your profile instead of **default**, in each instruction.
Similarly, substitute your profile for the **default** profile in the example CLI commands.
- ▶ If you use a Standalone Server, only one profile exists. Ignore any instructions to choose a specific profile. In CLI commands, remove the **/profile=default** portion of the sample commands.



Note

In order for the TM options to be visible in the Management Console or Management CLI, the **transactions** subsystem must be enabled. It is enabled by default, and required for many other subsystems to function properly, so it is very unlikely that it would be disabled.

Configure the TM Using the Management Console

To configure the TM using the web-based Management Console, select the **Runtime** tab from the list in the upper left side of the Management Console screen. If you use a managed domain, you have the choice of several profiles. Choose the correct one from the **Profile** selection box at the upper right of the Profiles screen. Expand the **Container** menu and select **Transactions**.

Most options are shown in the Transaction Manager configuration page. The **Recovery** options are hidden by default. Click the **Recovery** header to expand them. Click the **Edit** button to edit any of the options. Changes take effect immediately.

Click the **Need Help?** label to display in-line help text.

Configure the TM using the Management CLI

In the Management CLI, you can configure the TM using a series of commands. The commands all begin with **/profile=default/subsystem=transactions/** for a managed domain with profile **default**, or **/subsystem=transactions** for a Standalone Server.

Table 15.1. TM Configuration Options

Option	Description	CLI Command
Enable Statistics	Whether to enable transaction statistics. These statistics can be viewed in the Management Console in the Subsystem Metrics section of the Runtime tab.	<code>/profile=default/subsystem=transactions/:write-attribute(name=enable-statistics,value=true)</code>
Enable TSM Status	Whether to enable the transaction status manager (TSM) service, which is used for out-of-process recovery.	<code>/profile=default/subsystem=transactions/:write-attribute(name=enable-tsm-status,value=false)</code>
Default Timeout	The default transaction timeout. This defaults to 300 seconds. You can override this programmatically, on a per-transaction basis.	<code>/profile=default/subsystem=transactions/:write-attribute(name=default-timeout,value=300)</code>
Path	The relative or absolute filesystem path where the transaction manager core stores data. By default the value is a path relative to the value of the relative-to attribute.	<code>/profile=default/subsystem=transactions/:write-attribute(name=path,value=var)</code>
Relative To	References a global path configuration in the domain model. The default value is the data directory for JBoss Enterprise Application Platform 6, which is the value of the property <code>jboss.server.data.dir</code> , and defaults to <code>EAP_HOME/domain/data/</code> for a Managed Domain, or <code>EAP_HOME/standalone/data/</code> for a Standalone Server instance. The value of the path TM attribute is relative to this path. Use an empty string to disable the default behavior and force the value of the path attribute to be treated as an absolute path.	<code>/profile=default/subsystem=transactions/:write-attribute(name=relative-to,value=jboss.server.data.dir)</code>
Object Store Path	A relative or absolute filesystem path where the TM object store stores data. By default relative to the object-store-relative-to parameter's value.	<code>/profile=default/subsystem=transactions/:write-attribute(name=object-store-path,value=tx-object-store)</code>
Object Store Path Relative To	References a global path configuration in the domain model. The default value is the data directory for JBoss Enterprise Application Platform 6, which is the value of the property <code>jboss.server.data.dir</code> , and defaults to <code>EAP_HOME/domain/data/</code> for a Managed Domain, or <code>EAP_HOME/standalone/data/</code> for a Standalone Server instance. The value of the path	<code>/profile=default/subsystem=transactions/:write-attribute(name=object-store-relative-to,value=jboss.server.data.dir)</code>

	<p>TM attribute is relative to this path. Use an empty string to disable the default behavior and force the value of the path attribute to be treated as an absolute path.</p>	
Socket Binding	<p>Specifies the name of the socket binding used by the Transaction Manager for recovery and generating transaction identifiers, when the socket-based mechanism is used. Refer to process-id-socket-max-ports for more information on unique identifier generation. Socket bindings are specified per server group in the Server tab of the Management Console.</p>	<code>/profile=default/subsystem=transactions/:write-attribute(name=socket-binding,value=txn-recovery-environment)</code>
Status Socket Binding	Specifies the socket binding to use for the Transaction Status manager.	<code>/profile=default/subsystem=transactions/:write-attribute(name=status-socket-binding,value=txn-status-manager)</code>
Recovery Listener	Whether or not the Transaction Recovery process should listen on a network socket. Defaults to false .	<code>/profile=default/subsystem=transactions/:write-attribute(name=recovery-listener,value=false)</code>

The following options are for advanced use and can only be modified using the Management CLI. Be cautious when changing them from the default configuration. Contact Red Hat Global Support Services for more information.

Table 15.2. Advanced TM Configuration Options

Option	Description	CLI Command
jts	Whether to use Java Transaction Service (JTS) transactions. Defaults to false , which uses JTA transactions only.	<code>/profile=default/subsystem=transactions/:write-attribute(name=jts,value=false)</code>
node-identifier	The node identifier for the JTS service. This should be unique per JTS service, because the Transaction Manager uses this for recovery.	<code>/profile=default/subsystem=transactions/:write-attribute(name=node-identifier,value=1)</code>
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.</p> <p>The process-id-socket-max-ports represents the maximum number of sockets the TM will try before failing. The default value is 10.</p>	<code>/profile=default/subsystem=transactions/:write-attribute(name=process-id-socket-max-ports,value=10)</code>
process-id-uuid	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 . Refer to process-id-socket-max-ports for more information.	<code>/profile=default/subsystem=transactions/:write-attribute(name=process-id-uuid,value=true)</code>
use-hornetq-store	Use HornetQ's 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. If you enable this, also change the log-store value to hornetq .	<code>/profile=default/subsystem=transactions/:write-attribute(name=use-hornetq-store,value=false)</code>
log-store	Set to default if you use the file-system-based object store, or hornetq if you use the HornetQ journaling storage mechanism. If you set this to hornetq , also set use-hornetq-store to true .	<code>/profile=default/subsystem=transactions/log-store=log-store/:write-attribute(name=type,value=default)</code>

[Report a bug](#)

15.1.3. Configure Your Datasource to Use JTA Transactions

Task Summary

This task shows you how to enable Java Transactions API (JTA) on your datasource.

Task Prerequisites

You must meet the following conditions before continuing with this task:

- ▶ Your database or other resource must support JTA. If in doubt, consult the documentation for your database or other resource.
- ▶ Create a datasource. Refer to [Section 6.3.1, “Create a Non-XA Datasource with the Management Interfaces”](#).
- ▶ Stop the JBoss Enterprise Application Platform.
- ▶ Have access to edit the configuration files directly, in a text editor.

Procedure 15.1. Task:

1. **Open the configuration file in a text editor.**

Depending on whether you run the JBoss Enterprise Application Platform in a managed domain or standalone server, your configuration file will be in a different location.

A. **Managed domain**

The default configuration file for a managed domain is in
EAP_HOME/domain/configuration/domain.xml for Red Hat Enterprise Linux, and
EAP_HOME\domain\configuration\domain.xml for Microsoft Windows Server.

B. **Standalone server**

The default configuration file for a standalone server is in
EAP_HOME/standalone/configuration/standalone.xml for Red Hat Enterprise Linux,
and **EAP_HOME\standalone\configuration\standalone.xml** for Microsoft Windows Server.

2. **Locate the <datasource> tag that corresponds to your datasource.**

The datasource will have the **jndi-name** attribute set to the one you specified when you created it. For example, the ExampleDS datasource looks like this:

```
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="H2DS"
enabled="true" jta="true" use-java-context="true" use-ccm="true">
```

3. **Set the jta attribute to true.**

Add the following to the contents of your **<datasource>** tag, as they appear in the previous step:
jta="true"

4. **Save the configuration file.**

Save the configuration file and exit the text editor.

5. **Start the JBoss Enterprise Application Platform.**

Relaunch the JBoss Enterprise Application Platform 6 server.

Result:

The JBoss Enterprise Application Platform starts, and your datasource is configured to use JTA transactions.

[Report a bug](#)

15.1.4. Configure an XA Datasource

Task Prerequisites:

In order to add an XA Datasource, you need to log into the Management Console. See [Section 3.2.2, “Log in to the Management Console”](#) for more information.

1. **Add a new datasource.**

Add a new datasource to the JBoss Enterprise Application Platform. Follow the instructions in [Section 6.3.1, “Create a Non-XA Datasource with the Management Interfaces”](#), but click the **XA Datasource** tab at the top.

2. **Configure additional properties as appropriate.**

All datasource parameters are listed in [Section 6.6.1, “Datasource Parameters”](#).

Result:

Your XA Datasource is configured and ready to use.

[Report a bug](#)

15.1.5. About Transaction Log Messages

To track transaction status while keeping the log files readable, use the **DEBUG** log level for the transaction logger. For detailed debugging, use the **TRACE** log level. Refer to [Section 15.1.6, “Configure Logging for the Transaction Subsystem”](#) for information on configuring the transaction logger.

The transaction manager can generate a lot of logging information when configured to log in the **TRACE** log level. Following are some of the most commonly-seen messages. This list is not comprehensive, so you may see other messages than these.

Table 15.3. Transaction State Change

Transaction Begin	When a transaction begins the following code is executed: <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction::Begin:1342</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>tsLogger.logger.trace("BasicAction::Begin() for action-id "+ get_uid());</pre> </div>
Transaction Commit	When a transaction commits the following code is executed: <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction::End:1342</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>tsLogger.logger.trace("BasicAction::End() for action-id "+ get_uid());</pre> </div>
Transaction Rollback	When a transaction commits the following code is executed: <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction::Abort:1575</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>tsLogger.logger.trace("BasicAction::Abort() for action-id "+ get_uid());</pre> </div>
Transaction Timeout	When a transaction times out the following code is executed: <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>com.arjuna.ats.arjuna.coordinator.TransactionReaper::doCancellations:349</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <pre>tsLogger.logger.trace("Reaper Worker "+ Thread.currentThread() + " attempting to cancel "+ e._control.get_uid());</pre> </div> <div style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;"> <p>You will then see the same thread rolling back the transaction as shown above</p> </div>

[Report a bug](#)

15.1.6. Configure Logging for the Transaction Subsystem

Task Summary

Use this procedure to control the amount of information logged about transactions, independent of other logging settings in the JBoss Enterprise Application Platform. The main procedure shows how to do this in the web-based Management Console. The Management CLI command is given afterward.

Procedure 15.2. Configure the Transaction Logger Using the Management Console

1. Navigate to the Logging configuration area.

In the Management Console, click the **Profiles** tab at the top left of the screen. If you use a managed domain, choose the server profile you wish to configure, from the **Profile** selection box at the top right.

Expand the **Core** menu, and click the **Logging** label.

2. Edit the `com.arjuna` attributes.

Click the **Edit** button in the **Details** section, toward the bottom of the page. This is where you can add class-specific logging information. The `com.arjuna` class is already present. You can change the log level and whether to use parent handlers.

Log Level

The log level is **WARN** by default. Because transactions can produce a large quantity of logging output, the meaning of the standard logging levels is slightly different for the transaction logger. In general, messages tagged with levels at a lower severity than the chosen level are discarded.

Transaction Logging Levels, from Most to Least Verbose

- ▶ DEBUG
- ▶ INFO
- ▶ WARN
- ▶ ERROR
- ▶ FAILURE

Use Parent Handlers

Whether the logger should send its output to its parent logger. The default behavior is **true**.

3. Changes take effect immediately.

[Report a bug](#)

15.2. Transaction Administration

15.2.1. Browse and Manage Transactions

The command-line based Management CLI supports the ability to browse and manipulate transaction records. This functionality is provided by the interaction between the Transaction Manager and the Management API of JBoss Enterprise Application Platform 6.

The transaction manager stores information about each pending transaction and the participants involved the transaction, in a persistent storage called the *object store*. The Management API exposes the object store as a resource called the **log-store**. An API operation called **probe** reads the transaction logs and creates a node for each log. You can call the **probe** command manually, whenever you need to refresh the **log-store**. It is normal for transaction logs to appear and disappear quickly.

Example 15.1. Refresh the Log Store

This command refreshes the Log Store for server groups which use the profile **default** in a managed domain. For a standalone server, remove the **profile=default** from the command.

```
/profile=default/subsystem=transactions/log-store=log-store/:probe
```

Example 15.2. View All Prepared Transactions

To view all prepared transactions, first refresh the log store (see [Example 15.1, “Refresh the Log Store”](#)), then run the following command, which functions similarly to a filesystem `ls` command.

```
ls /profile=default/subsystem=transactions/log-store=log-store/transactions
```

Each transaction is shown, along with its unique identifier. Individual operations can be run against an individual transaction (see [Manage a Transaction](#)).

Manage a Transaction

View a transaction's attributes.

To view information about a transaction, such as its JNDI name, EIS product name and version, or its status, use the `:read-resource` CLI command.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:\ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-resource
```

View the participants of a transaction.

Each transaction log contains a child element called **participants**. Use the `:read-resource` CLI command on this element to see the participants of the transaction. Participants are identified by their JNDI names.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:\ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=java\:\JmsXA:read-resource
```

The result may look similar to this:

```
{
  "outcome" => "success",
  "result" => {
    "eis-product-name" => "HornetQ",
    "eis-product-version" => "2.0",
    "jndi-name" => "java:/JmsXA",
    "status" => "HEURISTIC",
    "type" => "/StateManager/AbstractRecord/XAResourceRecord"
  }
}
```

The outcome status shown here is in a **HEURISTIC** state and is eligible for recover. Refer to [Recover a transaction](#), for more details.

Delete a transaction.

Each transaction log supports a `:delete` operation, to delete the transaction log representing the transaction.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:\ffff7f000001\:-b66efc2\:4f9e6f8f\:9:delete
```

Recover a transaction.

Each transaction log supports recovery via the `:recover` CLI command.

Recovery of Heuristic Transactions and Participants

- ▶ If the transaction's status is **HEURISTIC**, the recovery operation changes the state to **PREPARE** and triggers a recovery.
- ▶ If one of the transaction's participants is heuristic, the recovery operation tries to reply the **commit** operation. If successful, the participant is removed from the transaction log. You can verify this by re-running the `:probe` operation on the **log-store** and checking that the participant is no longer listed. If this is the last participant, the transaction is also deleted.

Refresh the status of a transaction which needs recovery.

If a transaction needs recovery, you can use the **:refresh** CLI command to be sure it still requires recovery, before attempting the recovery.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:\fffff7f000001\:-b66efc2\:\4f9e6f8f\:\9:refresh
```

 **Note: Browsing JTS Transactions**

For JTS transactions, if participants are on remote servers, a limited amount of information may be available to the Transaction Manager. In this case, it is recommended that you use the file-based object store, rather than the HornetQ storage mode. This is the default behavior. To use the HornetQ storage mode, you can set the value of the **use-hornetq-store** option to **true**, in the Transaction Manager configuration. Refer to [Section 15.1.3, “Configure Your Datasource to Use JTA Transactions”](#) for information on configuring the Transaction Manager.

View Transaction Statistics

You can view statistics about the Transaction Manager and transaction subsystem either via the web-based Management Console or the command-line Management CLI.

In the web-based Management Console, Transaction statistics are available via **Runtime → Subsystem Metrics → Transactions**. Transaction statistics are available for each server in a managed domain, as well. You can specify the server in the **Server** selection box at the top left.

The following table shows each available statistic, its description, and the CLI command to view the statistic.

Table 15.4. Transaction Subsystem Statistics

Statistic	Description	CLI Command
Total	The total number of transactions processed by the Transaction Manager on this server.	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-transactions,include-defaults=true)</pre>
Committed	The number of committed transactions processed by the Transaction Manager on this server.	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-committed-transactions,include-defaults=true)</pre>
Aborted	The number of aborted transactions processed by the Transaction Manager on this server.	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-aborted-transactions,include-defaults=true)</pre>
Timed Out	The number of timed out transactions processed by the Transaction Manager on this server.	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-timed-out-transactions,include-defaults=true)</pre>
Heuristics	Not available in the Management Console. Number of transactions in a heuristic state.	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-heuristics,include-defaults=true)</pre>
In-Flight Transactions	Not available in the Management Console. Number of transactions which have begun but not yet terminated.	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-inflight-transactions,include-defaults=true)</pre>
Failure Origin - Applications	The number of failed	

Failure Origin - Applications	The number of failed transactions whose failure origin was an application.	<code>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-application-rollback, include-defaults=true)</code>
Failure Origin - Resources	The number of failed transactions whose failure origin was a resource.	<code>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-resource-rollback, include-defaults=true)</code>

[Report a bug](#)

15.3. Transaction References

15.3.1. JBoss Transactions Errors and Exceptions

For details about exceptions thrown by methods of the `UserTransaction` class, see the *UserTransaction API* specification at

<http://download.oracle.com/javaee/1.3/api/javax/transaction/UserTransaction.html>.

[Report a bug](#)

15.3.2. JTA Clustering Limitations

JTA transactions cannot be clustered across multiple instances of the JBoss Enterprise Application Platform. For this behavior, use JTS transactions.

To use JTS transactions, you need to configure the ORB: [Section 15.4.2, "Configure the ORB for JTS Transactions"](#).

[Report a bug](#)

15.4. ORB Configuration

15.4.1. 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 Enterprise Application Platform 6 provides an ORB instance, by means of the JacORB component.

The ORB is used internally for Java Transaction Service (JTS) transactions, and is also available for use by your own applications.

[Report a bug](#)

15.4.2. Configure the ORB for JTS Transactions

In a default installation of JBoss Enterprise Application Platform, the ORB is disabled. You can enable the ORB using the command-line Management CLI.

**Note**

In a managed domain, the JacORB subsystem is available in **full** and **full-ha** profiles only. In a standalone server, it is available when you use the **standalone-full.xml** or **standalone-full-ha.xml** configurations.

Procedure 15.3. Configure the ORB using the Management Console**1. View the profile settings.**

Select **Profiles** (managed domain) or **Profile** (standalone server) from the top right of the management console. If you use a managed domain, select either the **full** or **full-ha** profile from the selection box at the top left.

2. Modify the Initializers Settings

Expand the **Subsystems** menu at the left, if necessary. Expand the **Container** sub-menu and click **JacORB**.

In the form that appears in the main screen, select the **Initializers** tab and click the **Edit** button.

Enable the security interceptors by setting the value of **Security** to **on**.

To enable the ORB for JTS, set the **Transaction Interceptors** value to **on**, rather than the default **spec**.

Refer to the **Need Help?** link in the form for detailed explanations about these values. Click **Save** when you have finished editing the values.

3. Advanced ORB Configuration

Refer to the other sections of the form for advanced configuration options. Each section includes a **Need Help?** link with detailed information about the parameters.

Configure the ORB using the Management CLI

You can configure each aspect of the ORB using the Management CLI. The following commands configure the initializers to the same values as the procedure above, for the Management Console. This is the minimum configuration for the ORB to be used with JTS.

These commands are configured for a managed domain using the **full** profile. If necessary, change the profile to suit the one you need to configure. If you use a standalone server, omit the **/profile=full** portion of the commands.

Example 15.3. Enable the Security Interceptors

```
/profile=full/subsystem=jacorb/:write-attribute(name=security,value=on)
```

Example 15.4. Enable the ORB for JTS

```
/profile=full/subsystem=jacorb/:write-attribute(name=transactions,value=on)
```

[Report a bug](#)

Chapter 16. Enterprise JavaBeans

16.1. Introduction

16.1.1. Overview of Enterprise JavaBeans

Enterprise JavaBeans (EJB) 3.1 is an API for developing distributed, transactional, secure and portable Java EE applications through the use of server-side components called Enterprise Beans. Enterprise Beans implement the business logic of an application in a decoupled manner that encourages reuse. Enterprise JavaBeans 3.1 is documented as the Java EE specification JSR-318.

JBoss Enterprise Application Platform 6 has full support for applications built using the Enterprise JavaBeans 3.1 specification. The EJB Container is implemented using the JBoss EJB3 community project, <http://www.jboss.org/ejb3>.

[Report a bug](#)

16.1.2. Overview of Enterprise JavaBeans for Administrators

JBoss administrators have many configuration options available to them to control the performance of Enterprise Beans in JBoss Enterprise Application Platform 6. These options can be accessed using the Management Console or the command line configuration tool. Editing the XML server configuration file to apply changes is also possible but not recommended.

The EJB configuration options are located in slightly different places in the Management Console depending on how the server is being run.

If the server is running as a Standalone Server:

1. Clicking on the **Profile** link on the top right to switch to the Profile view.
2. Expand the **Profile** menu on the left by clicking the arrow next to the label.
3. Click on **Container** to expand it, and then click on **EJB 3**.

If the server is running a part of a Managed Domain:

1. Click on the **Profile** link on the top to switch to the Profile view.
2. Expand the **Subsystems** menu on the left by clicking the arrow next to the label.
3. Select the profile you are modifying from the **Profile** menu.
4. Click on **Container** to expand it, and then click on **EJB 3**.

The screenshot shows the JBoss Enterprise Application Platform 6 Management Console. The left sidebar is a navigation tree with sections like Profile, Connector, Container, Transactions, Naming, EJB 3 (which is selected and highlighted in grey), EE, JPA, Core, Infinispan, OSGi, Security, and Web. Under EJB 3, there's a 'General Configuration' section with 'Interfaces'. The main content area is titled 'EJB3 Container Configuration' and contains a note about overriding settings at deployment or bean level. It shows three configuration sections: 'Stateless Session Pool: slsb-strict-max-pool' (with 'Edit' and 'Need Help?' buttons), 'Default Resource Adapter: hornetq-ra' (with 'Edit' and 'Need Help?' buttons), and 'Message Driven Pool' (with 'Edit' and 'Need Help?' buttons). At the bottom of the content area, there are 'Settings' and 'Logout' links. The footer of the page shows the version '1.3.1.Final-redhat-1'.

Figure 16.1. EJB Configuration Options in the Management Console (Standalone Server)

[Report a bug](#)

16.1.3. Enterprise Beans

Enterprise beans are server-side application components as defined in the Enterprise JavaBeans (EJB) 3.1 specification, JSR-318. Enterprise beans are designed for the implementation of application business

logic in a decoupled manner to encourage reuse.

Enterprise beans are written as Java classes and annotated with the appropriate EJB annotations. They can be deployed to the application server in their own archive (a JAR file) or be deployed as part of a Java EE application. The application server manages the lifecycle of each enterprise bean and provides services to them such as security, transactions, and concurrency management.

An enterprise bean can also define any number of business interfaces. Business interfaces provide greater control over which of the bean's methods are available to clients and can also allow access to clients running in remote JVMs.

There are three types of Enterprise Bean: Session beans, Message-driven beans and Entity beans.



Important

Entity beans are now deprecated in EJB 3.1 and Red Hat recommends the use of JPA entities instead. Red Hat only recommends the use of Entity beans for backwards compatibility with legacy systems.

[Report a bug](#)

16.1.4. Session Beans

Session Beans are Enterprise Beans that encapsulate a set of related business processes or tasks and are injected into the classes that request them. There are three types of session bean: stateless, stateful, and singleton.

[Report a bug](#)

16.1.5. Message-Driven Beans

Message-driven Beans (MDBs) provide an event driven model for application development. The methods of MDBs are not injected into or invoked from client code but are triggered by the receipt of messages from a messaging service such as a Java Messaging Service (JMS) server. The Java EE 6 specification requires that JMS is supported but other messaging systems can be supported as well.

[Report a bug](#)

16.2. Configuring Bean Pools

16.2.1. Bean Pools

JBoss Enterprise Application Platform 6 maintains a number of instances of deployed stateless enterprise beans in memory to provide faster performance. This technique is called bean pooling. When a bean is required the application server can take one from the appropriate pool of already available beans instead of instantiating a new one. When the bean is no longer required it is returned to the pool for reuse.

Bean pools are configured and maintained separately for stateless session beans and for message-driven beans.

[Report a bug](#)

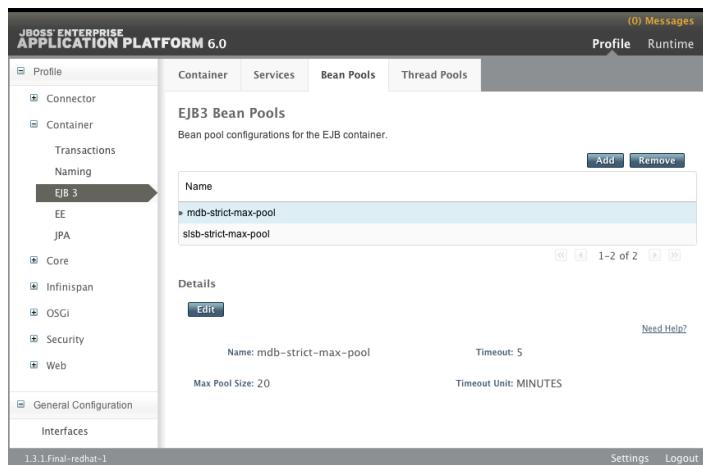
16.2.2. Create a Bean Pool

Bean pools can be created using the Management Console and the CLI tool.

Bean pools can also be created by adding the required bean pool configuration to the server configuration file using a text editor. [Example 16.2, "XML Configuration Sample"](#) is an example of what this configuration looks like.

Procedure 16.1. Create a bean pool using the Management Console

1. Login to the Management Console. Refer to [Section 3.2.2, "Log in to the Management Console"](#).
2. Navigate to the **EJB3 Bean Pools** panel.

**Figure 16.2. EJB3 Bean Pools Panel**

3. Click the **Add** button. The **Add EJB3 Bean Pools** dialog appears.
4. Specify the required details, **Name**, **Max Pool Size**, **Timeout** value, and **Timeout unit**.
5. Click on the **Save** button to save the new bean pool or click the **Cancel** link to abort the procedure.
 - » If you click the **Save** button, the dialog will close and the new bean pool will appear in the list.
 - » If you click **Cancel**, the dialog will close and no new bean pool will be created.

Procedure 16.2. Create a bean pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **add** operation with the following syntax.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:add(max-pool-size=MAXSIZE, timeout=TIMEOUT, timeout-unit="UNIT")
```

- » Replace **BEANPOOLNAME** with the required name for the bean pool.
- » Replace **MAXSIZE** with the maximum size of the bean pool.
- » Replace **TIMEOUT**
- » Replace **UNIT** with the required time unit. Allowed values are: **NANOSECONDS**, **MICROSECONDS**, **MILLISECONDS**, **SECONDS**, **MINUTES**, **HOURS**, and **DAYS**.

3. Use the **read-resource** operation to confirm the creation of the bean pool.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

Example 16.1. Create a Bean Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-pool=ACCTS_BEAN_POOL:add(max-pool-size=500, timeout=5000, timeout-unit="SECONDS")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Example 16.2. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">

<pools>
    <bean-instance-pools>
        <strict-max-pool name="slsb-strict-max-pool" max-pool-size="20"
            instance-acquisition-timeout="5"
            instance-acquisition-timeout-unit="MINUTES" />
        <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20"
            instance-acquisition-timeout="5"
            instance-acquisition-timeout-unit="MINUTES" />
    </bean-instance-pools>
</pools>
</subsystem>
```

[Report a bug](#)

16.2.3. Remove a Bean Pool

Unused bean pools can be removed using the Management Console.

Prerequisites:

- The bean pool that you want to remove cannot be in use. Refer to [Section 16.2.5, “Assign Bean Pools for Session and Message-Driven Beans”](#) to ensure that it is not being used.

Procedure 16.3. Remove a bean pool using the Management Console

- Login to the Management Console. Refer to [Section 3.2.2, “Log in to the Management Console”](#).
- Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Bean Pools** tab from the main panel.

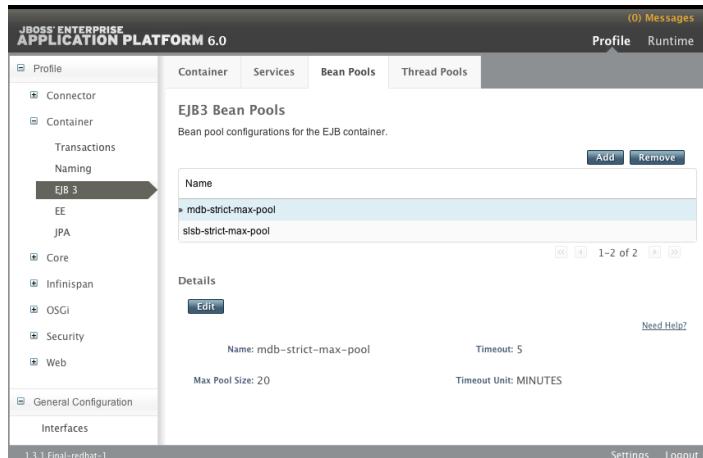


Figure 16.3. EJB3 Bean Pools Panel

- Select the bean pool to remove in the list.
- Click the **Remove** button. The **Remove Item** dialog appears.
- Click the **OK** button to confirm deletion or click the **Cancel** link to abort the operation.
If you click the **Ok** button, the dialog will close and the bean pool will be deleted and removed from the list.
If you click Cancel, the dialog will close and no changes will be made.

Procedure 16.4. Remove a bean pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **remove** operation with the following syntax.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:remove
```

» Replace **BEANPOOLNAME** with the required name for the bean pool.

Example 16.3. Removing a Bean Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-
pool=ACCTS_BEAN_POOL:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

16.2.4. Edit a Bean Pool

Bean pools can be edited using the Management Console.

Procedure 16.5. Edit a bean pool using the Management Console

1. Login to the Management Console. [Section 3.2.2, “Log in to the Management Console”](#)
2. Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Bean Pools** tab from the main panel.

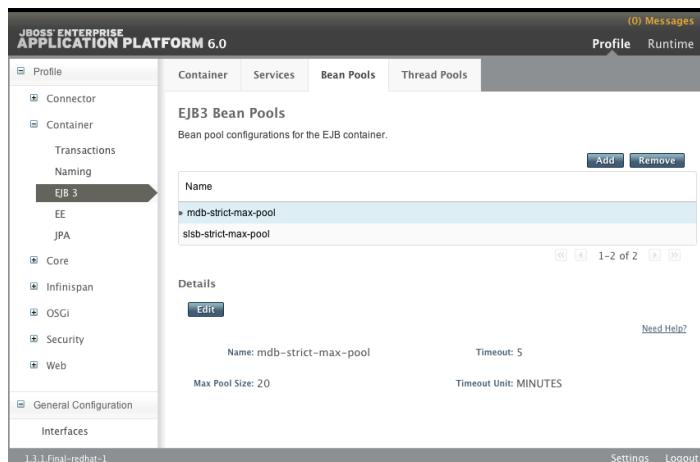


Figure 16.4. EJB3 Bean Pools Panel

3. Click on the bean pool you wish to edit from the list.
 4. Click the **Edit** button. The fields in the **Details** area are now editable.
 5. Edit the details you want to change. Only **Max Pool Size**, **Timeout** value, and **Timeout** unit can be changed.
 6. Click the **Save** button if you are satisfied with the changes, or click the **Cancel** link if you want to discard the changes.
- If you click the **Ok** button, the **Details** area will change back to its non-editable form and the bean pool will be updated with the new details.
- If you click the **Cancel** link, the **Details** area will change back to its non-editable form and no changes will be made.

Procedure 16.6. Edit a bean pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **write_attribute** operation with the following syntax for each attribute of the bean pool to be changed.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:write-
attribute(name="ATTRIBUTE", value="VALUE")
```

- ▶ Replace **BEANPOOLNAME** with the required name for the bean pool.
- ▶ Replace **ATTRIBUTE** with the name of the attribute to be edited. The attributes that can be edited in this way are **max-pool-size**, **timeout**, and **timeout-unit**.
- ▶ Replace **VALUE** with the required value of the attribute.

3. Use the **read-resource** operation to confirm the changes to the bean pool.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

Example 16.4. Set the Timeout Value of a Bean Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-
pool=HSBeanPool:write-attribute(name="timeout", value="1500")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

16.2.5. Assign Bean Pools for Session and Message-Driven Beans

JBoss Administrators can assign individual bean pools for use by session beans and message-driven beans. Bean pools can be assigned by using the Management Console or the CLI tool.

By default two bean pools are provided, **slsb-strict-max-pool** and **mdb-strict-max-pool** for stateless session beans and message-driven beans respectively.

To create or edit bean pools, refer to [Section 16.2.2, “Create a Bean Pool”](#) and [Section 16.2.4, “Edit a Bean Pool”](#).

Procedure 16.7. Assign Bean Pools for Session and Message-Driven Beans using the Management Console

1. Login to the Management Console. [Section 3.2.2, “Log in to the Management Console”](#)
2. Navigate to the EJB3 Container Configuration panel.

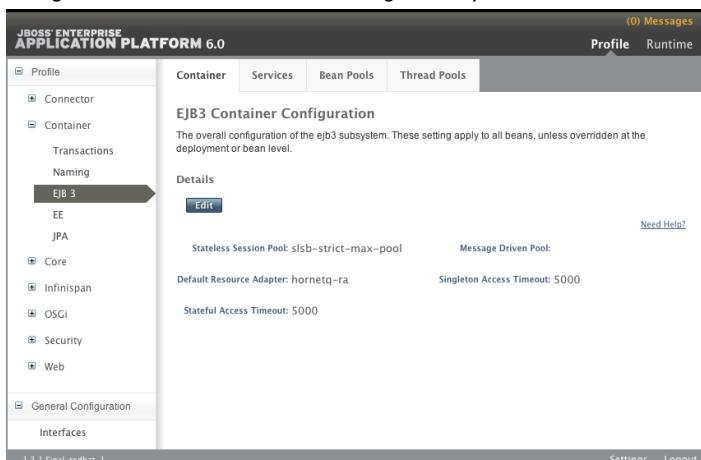


Figure 16.5. EJB3 Container Configuration panel in the Management Console (Standalone Server)

3. Click the **Edit** button. The fields in the **Details** area are now editable.
4. Select the bean pool to use for each type of bean from the appropriate combo-box.
5. Click the **Save** button to keep the changes, or click the **Cancel** link to discard them.
6. The Details area will now be non-editable and display the correct bean pool selection.

Procedure 16.8. Assign Bean Pools for Session and Message-Driven Beans using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, “Connect to a Managed](#)

[Server Instance Using the Management CLI](#).

2. Use the **write-attribute** operation with the following syntax.

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value="BEANPOOL")
```

- ▶ Replace **BEANTYPE** with **default-mdb-instance-pool** for Message-Driven Beans or **default-slsb-instance-pool** for stateless session beans.
- ▶ Replace **BEANPOOL** with the name of the bean pool to assign.

3. Use the **read-resource** operation to confirm the changes.

```
/subsystem=ejb3:read-resource
```

Example 16.5. Assign a Bean Pool for Session Beans using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-slsb-instance-pool", value="LV_SLSB_POOL")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Example 16.6. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>

</subsystem>
```

[Report a bug](#)

16.3. Configuring EJB Thread Pools

16.3.1. Enterprise Bean Thread Pools

JBoss Enterprise Application Platform 6 maintains number of instances of Java thread objects in memory for use by enterprise bean services, including remote invocation, the timer service, and asynchronous invocation.

This technique is called thread pooling. It provides improved performance by eliminating the overhead of thread creation and gives the system administrator a mechanism for controlling resource usage.

Multiple thread pools can be created with different parameters and each service can be allocated a different thread pool.

[Report a bug](#)

16.3.2. Create a Thread Pool

EJB Thread pools can be created using the Management Console or the CLI.

Procedure 16.9. Create an EJB Thread Pool using the Management Console

1. Login to the Management Console. [Section 3.2.2, “Log in to the Management Console”](#)
2. Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Thread Pools** tab from the main panel.

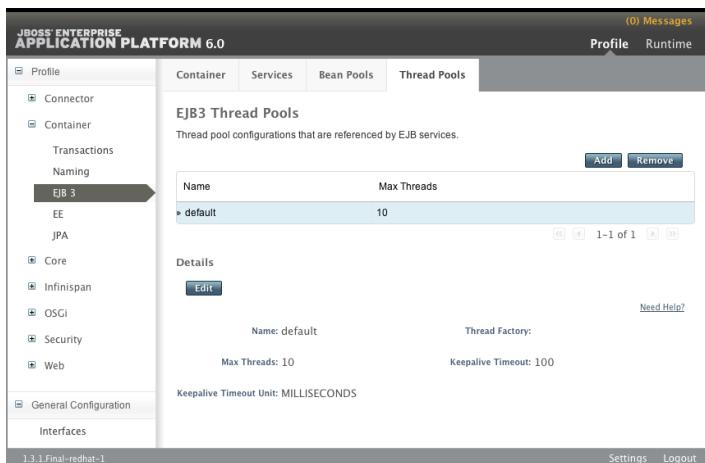


Figure 16.6. EJB3 Thread Pools Panel

3. Click the **Add** button. The **Add EJB3 Thread Pools** dialog appears.
4. Specify the required details, **Name**, **Max. Threads**, and **Keep-Alive Timeout** value.
5. Click on the **Save** button to save the new thread pool or click the **Cancel** link to abort the procedure.
 - » If you click the **Save** button, the dialog will close and the new thread pool will appear in the list.
 - » If you click **Cancel**, the dialog will close and no new thread pool will be created.

Procedure 16.10. Create a Thread Pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **add** operation with the following syntax.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:add(max-threads=MAXSIZE,
keepalive-time={"time"=>"TIME", "unit"=>UNIT})
```

- » Replace **THREADPOOLNAME** with the required name for the thread pool.
- » Replace **MAXSIZE** with the maximum size of the thread pool.
- » Replace **UNIT** with the required time unit to be used for the required keep-alive time. Allowed values are: **NANOSECONDS**, **MICROSECONDS**, **MILLISECONDS**, **SECONDS**, **MINUTES**, **HOURS**, and **DAYS**.
- » Replace **TIME** with the integer value of the required keep-alive time. This value is a number of **UNITS**.

3. Use the **read-resource** operation to confirm the creation of the bean pool.

```
/subsystem=ejb3/strict-max-bean-instance-pool=THREADPOOLNAME:read-resource
```

Example 16.7. Create a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=testmepool:add(max-
threads=50, keepalive-time={"time"=>"150", "unit"=>"SECONDS"})
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Example 16.8. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <thread-pools>
    <thread-pool name="default" max-threads="20" keepalive-time="150"/>
  </thread-pools>
</subsystem>
```

[Report a bug](#)

16.3.3. Remove a Thread Pool

Unused EJB thread pools can be removed using the Management Console.

Prerequisites

- » The thread pool that you want to remove cannot be in use. Refer to the following tasks to ensure that the thread pool is not in use:
 - [Section 16.6.2, “Configure the EJB3 timer Service”](#)
 - [Section 16.7.2, “Configure the EJB3 Asynchronous Invocation Service Thread Pool”](#)
 - [Section 16.8.2, “Configure the EJB3 Remote Service”](#)

Procedure 16.11. Remove an EJB thread pool using the Management Console

1. Login to the Management Console. [Section 3.2.2, “Log in to the Management Console”](#).
2. Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Thread Pools** tab from the main panel.

Name	Max Threads
default	10

Details

Name: default Thread Factory:
Max Threads: 10 Keepalive Timeout: 100
Keepalive Timeout Unit: MILLISECONDS

Figure 16.7. EJB3 Thread Pools Panel

3. Select the thread pool to remove in the list.
4. Click the **Remove** button. The **Remove Item** dialog appears.
5. Click the **OK** button to confirm deletion or click the **Cancel** link to abort the operation.
If you click the **Ok** button, the dialog will close and the thread pool will be deleted and removed from the list.
If you click **Cancel**, the dialog will close and no changes will be made.

Procedure 16.12. Remove a thread pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **remove** operation with the following syntax.

```
/subsystem=ejb3/thread-pool=<THREADPOOLNAME>:remove
```

- » Replace **<THREADPOOLNAME>** with the name of the thread pool.

Example 16.9. Removing a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=ACCTS_THREADS:remove
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

16.3.4. Edit a Thread Pool

JBoss Administrators can edit Thread Pools using the Management Console and the CLI.

Procedure 16.13. Edit a Thread Pool using the Management Console

1. Login

Login to the Management Console.

2. Navigate to the EJB3 Thread Pools Tab

Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Thread Pools** tab from the main panel.

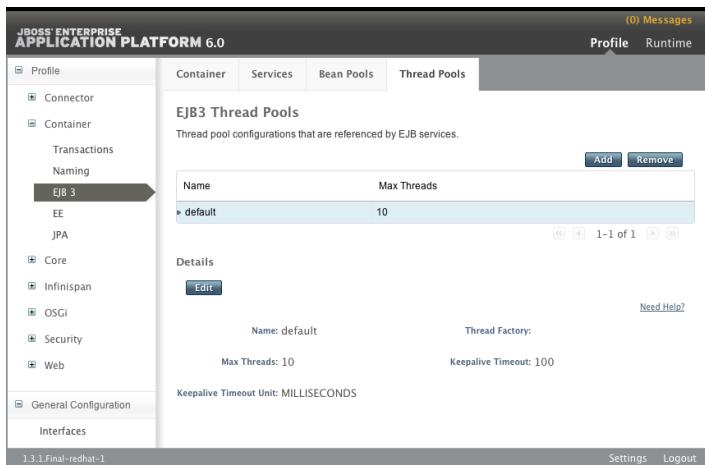


Figure 16.8. EJB3 Thread Pools Tab

3. Select the Thread Pool to Edit

Select the thread pool you wish to edit from the list.

4. Click the Edit button

The fields in the Details area are now editable.

5. Edit Details

Edit the details you want to change. Only the **Thread Factory**, **Max Threads**, **Keepalive Timeout**, and **Keepalive Timeout Unit** values can be edited.

6. Save or Cancel

Click the **Save** button if you are satisfied with the changes, or click the **Cancel** link if you want to discard the changes.

Procedure 16.14. Edit a thread pool using the CLI

- Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#).

- Use the **write_attribute** operation with the following syntax for each attribute of the thread pool to be changed.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-attribute(name="ATTRIBUTE", value="VALUE")
```

- » Replace **THREADPOOLNAME** with the name of the thread pool.
- » Replace **ATTRIBUTE** with the name of the attribute to be edited. The attributes that can be edited in this way are **keepalive-time**, **max-threads**, and **thread-factory**.
- » Replace **VALUE** with the required value of the attribute.

- Use the **read-resource** operation to confirm the changes to the thread pool.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:read-resource
```

**Important**

When changing the value of the **keepalive-time** attribute with the CLI the required value is an object representation. It has the following syntax.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-attribute(name="keepalive-time", value={"time" => "VALUE", "unit" => "UNIT"})
```

Example 16.10. Set the Maxsize Value of a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-attribute(name="max-threads", value="50")
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

Example 16.11. Set the keepalive-time Time Value of a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-attribute(name="keepalive-time", value={"time"=>"150"})
 {"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

16.4. Configuring Session Beans

16.4.1. Session Bean Access Timeout

Stateful and Singleton Session Beans have an access timeout value specified for managing concurrent access. This value is the period of time that a request to a session bean method can be blocked before it will timeout.

The timeout value and the time unit used can be specified using the **@javax.ejb.AccessTimeout** annotation on the method. It can be specified on the session bean (which applies to all the bean's methods) and on specific methods to override the configuration for the bean.

If they are not specified JBoss Enterprise Application Platform 6 supplies a default timeout value of 5000 milliseconds.

Refer to the Javadocs for AccessTimeout at
<http://docs.oracle.com/javaee/6/api/javax/ejb/AccessTimeout.html>

[Report a bug](#)

16.4.2. Set Default Session Bean Access Timeout Values

JBoss Administrators can specify the default timeout values for Singleton and Stateful session beans. The default timeout values can be changed using the Management Console or the CLI. The default value is 5000 milliseconds.

Procedure 16.15. Set Default Session Bean Access Timeout Values using the Management Console

1. Login to the Management Console. See [Section 3.2.2, “Log in to the Management Console”](#).
2. Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Container** tab from the main panel.

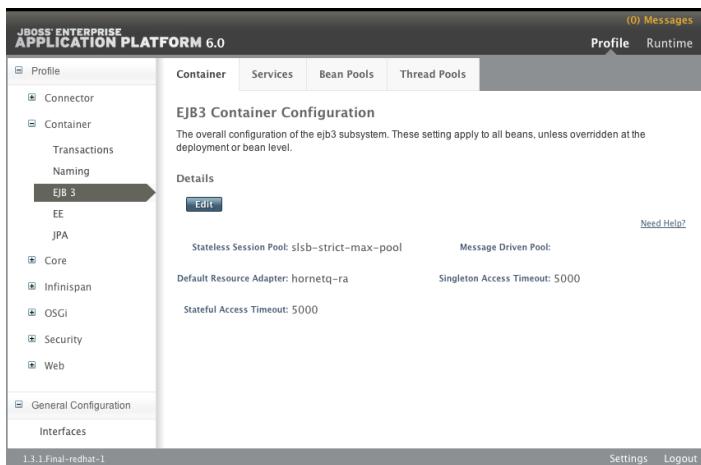


Figure 16.9. EJB3 Container Configuration panel in the Management Console (Standalone Server)

3. Click the **Edit** button. The fields in the **Details** area are now editable.
4. Enter the required values in the **Stateful Access Timeout** and/or **Singleton Access Timeout** text boxes.
5. Click the **Save** button to keep the changes, or click the **Cancel** link to discard them.
6. The **Details** area will now be non-editable and display the correct timeout values.

Procedure 16.16. Set Session Bean Access Timeout Values Using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **write-attribute** operation with the following syntax.

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value=TIME)
```

- ▶ Replace **BEANTYPE** with **default-stateful-bean-access-timeout** for Stateful Session Beans, or **default-singleton-bean-access-timeout** for Singleton Session Beans.
- ▶ Replace **TIME** with the required timeout value.

3. Use the **read-resource** operation to confirm the changes.

```
/subsystem=ejb3:read-resource
```

Example 16.12. Setting the Default Stateful Bean Access Timeout value to 9000 with the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-stateful-bean-access-timeout", value=9000)
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

Example 16.13. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>

</subsystem>
```

[Report a bug](#)

16.5. Configuring Message-Driven Beans

16.5.1. Set Default Resource Adapter for Message-Driven Beans

JBoss Administrators can specify the default resource adapter used by message-driven beans. The default resource adapter can be specified using the Management Console and the CLI. The default resource adapter supplied with JBoss Enterprise Application Platform 6 is **hornetq-ra**.

Procedure 16.17. Set the Default Resource Adapter for Message-Driven Beans using the Management Console

1. Login to the Management Console. [Section 3.2.2. "Log in to the Management Console"](#)
2. Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Then select the **Container** tab from the main panel.

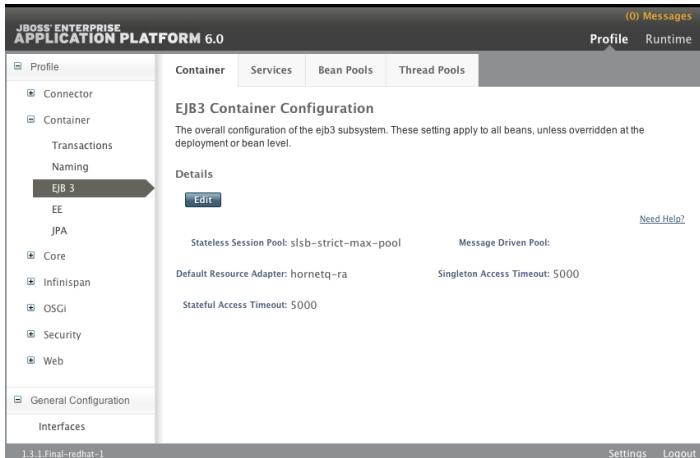


Figure 16.10. EJB3 Container Configuration panel in the Management Console (Standalone Server)

3. Click the **Edit** button. The fields in the **Details** area are now editable.
4. Enter the name of the resource adapter to be used in the **Default Resource Adapter** text box.
5. Click the **Save** button to keep the changes, or click the **Cancel** link to discard them.
6. The **Details** area will now be non-editable and display the correct resource adapter name.

Procedure 16.18. Set the Default Resource Adapter for Message-Driven Beans using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.3.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **write-attribute** operation with the following syntax.

```
/subsystem=ejb3:write-attribute(name="default-resource-adapter-name",
value="RESOURCE-ADAPTER")
```

Replace **RESOURCE-ADAPTER** with name of the resource adapter to be used.

3. Use the **read-resource** operation to confirm the changes.

```
/subsystem=ejb3:read-resource
```

Example 16.14. Set the Default Resource Adapter for Message-Driven Beans using the CLI

```
[standalone@localhost:9999 subsystem=ejb3] /subsystem=ejb3:write-
attribute(name="default-resource-adapter-name", value="EDIS-RA")
{"outcome" => "success"}
[standalone@localhost:9999 subsystem=ejb3]
```

Example 16.15. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>

</subsystem>
```

[Report a bug](#)

16.6. Configuring the EJB3 Timer Service

16.6.1. EJB3 Timer Service

The EJB3 Timer Service is a standard Java EE 6 service for scheduling the invocation of the methods from enterprise beans. Stateless session beans, singleton session beans, and message-driven beans can all schedule any of their methods for callback at specified times. Method callback can occur at a specific time, after a duration, at a recurring interval, or on a calendar-based schedule.

[Report a bug](#)

16.6.2. Configure the EJB3 timer Service

JBoss Administrators can configure the EJB3 Timer Service in the JBoss Enterprise Application Platform 6 Management Console. The features that can be configured are the thread pool that is used for scheduled bean invocation and the directory where the Timer Service data is stored.

Procedure 16.19. Configure the EJB3 Timer Service

1. Login

Login to the Management Console.

2. Open the Timer Service Tab

Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Select the **Services** tab from the main panel and then the **Timer Service** tab.

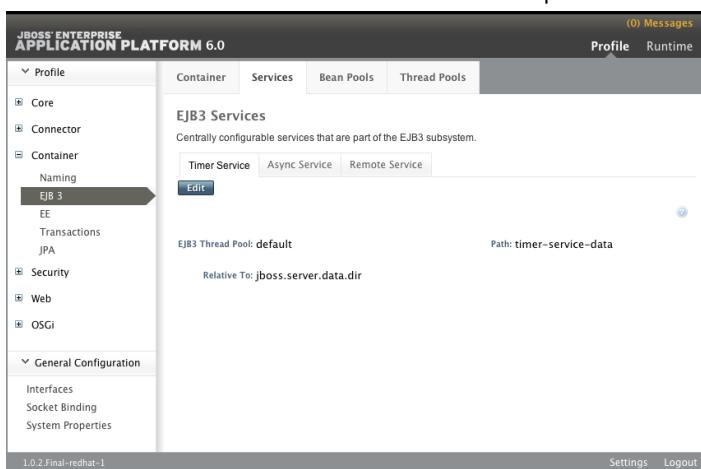


Figure 16.11. Timer Service tab of the EJB3 Services panel

3. Enter Edit Mode

Click the Edit Button. The fields become editable.

4. Make the Required Changes.

You can select a different EJB3 thread pool used for the Timer Service if additional thread pools have been configured, and you can change the directory used to save the Timer Service data.

The Timer Service data directory configuration consists of two values: **Path**, the directory that data is stored in; and **Relative To**, the directory which contains **Path**. By default **Relative To** is set to a Filesystem Path Variable.

5. Save or Cancel

Click the **Save** button to keep the changes, or click the **Cancel** link to discard them.

[Report a bug](#)

16.7. Configuring the EJB Asynchronous Invocation Service

16.7.1. EJB3 Asynchronous Invocation Service

The Asynchronous Invocation Service is an Enterprise JavaBeans container service that manages asynchronous invocation of session bean methods. This service maintains a configurable number of threads (a thread pool) that are allocated for asynchronous method execution.

Enterprise JavaBeans 3.1 allows for any method of a session bean (stateful, stateless or singleton) to be annotated to permit asynchronous execution.

[Report a bug](#)

16.7.2. Configure the EJB3 Asynchronous Invocation Service Thread Pool

JBoss Administrators can configure the EJB3 Asynchronous Invocation Service in the JBoss Enterprise Application Platform 6 Management Console to use a specific thread pool.

Procedure 16.20. Configure the EJB3 Asynchronous Invocation Service thread pool

1. Login

Login to the Management Console.

2. Open the Async Service tab

Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Select the **Services** tab from the main panel and then the **Async Service** tab.

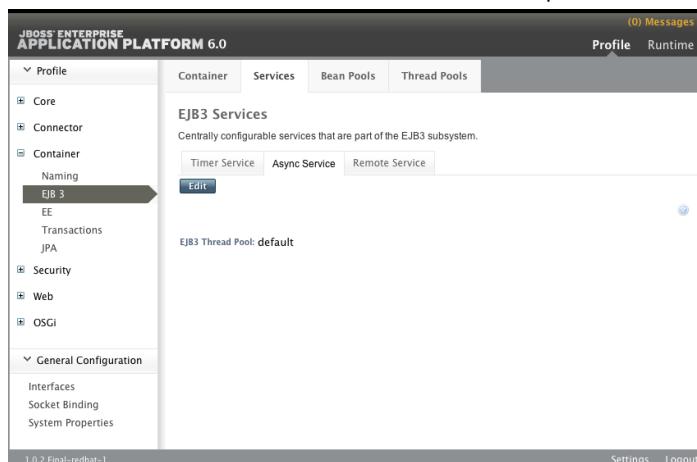


Figure 16.12. The Async Service tab of the EJB3 Services panel

3. Enter Edit Mode

Click the **Edit** Button. The fields become editable.

4. Select the thread pool

Select the EJB3 thread pool to use from the list. The thread pool must have been already created.

5. Save or Cancel

Click the **Save** button to keep the changes, or click the **Cancel** link to discard them.

[Report a bug](#)

16.8. Configuring the EJB3 Remote Invocation Service

16.8.1. EJB3 Remote Service

The EJB3 Remote Service manages the remote execution of Enterprise Beans with remote business interfaces.

[Report a bug](#)

16.8.2. Configure the EJB3 Remote Service

JBoss Administrators can configure the EJB3 Remote Service in the JBoss Enterprise Application Platform 6 Management Console. The features that can be configured are the thread pool that is used for remote bean invocation and the connector on which the EJB3 remoting channel is registered.

Procedure 16.21. Configure the EJB3 Remote Service

1. Login

Login to the Management Console.

2. Open the Remote Service tab

Click on **Profile** in the top right, expand the **Container** item in the Profile panel on the left and select **EJB 3**. Select the **Services** tab from the main panel, and then the **Remote Service** tab.

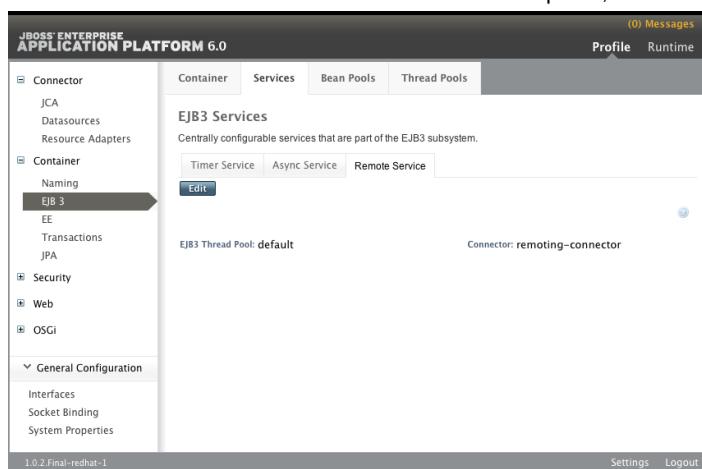


Figure 16.13. The Remote Service tab of the EJB3 Services panel

3. Enter Edit Mode

Click the **Edit** Button. The fields become editable.

4. Make the required changes

You can select a different EJB3 thread pool used for the Remote Service if additional thread pools have been configured. You can change the connector used to register the EJB remoting channel.

5. Save or Cancel

Click the **Save** button to keep the changes, or click the **Cancel** link to discard them.

[Report a bug](#)

16.9. Configuring EJB 2.x Entity Beans

16.9.1. EJB Entity Beans

EJB Entity Beans are a type of enterprise bean from version 2.x of the EJB specification that represented persistent data that was maintained in a database. Entity beans have been superseded by JPA entities and officially listed for removal (pruning) from future versions of the specification. Red Hat does not recommend the use of Entity Beans except for backwards compatibility.

Support for Entity Beans is disabled by default in JBoss Enterprise Application Platform 6.

[Report a bug](#)

16.9.2. Container-Managed Persistence

Container-Managed Persistence (CMP) is an application server provided service that provides data persistence for Entity beans.

[Report a bug](#)

16.9.3. Enable EJB 2.x Container-Managed Persistence

Container-Managed Persistence (CMP) is handled by the `org.jboss.as.cmp` extension. CMP is enabled by default in the managed domain and standalone server full configurations, e.g. `standalone-full.xml`.

To enable CMP in a different configuration, add the `org.jboss.as.cmp` module to the list of enabled extensions in the server configuration file.

```
<extensions>
    <extension module="org.jboss.as.cmp"/>
</extensions>
```

To disable CMP in a server configuration, remove the extension entry for the `org.jboss.as.cmp` module.

[Report a bug](#)

16.9.4. Configure EJB 2.x Container-Managed Persistence

The EJB 2.x Container Managed Persistence (CMP) subsystem can be configured to specify any number of key generators. Key generators are used to generate unique keys to identify each entity persisted by the CMP service.

Two types of key generator can be defined: UUID-based and HiLo key generators.

UUID-based key generators

A UUID-based key generator creates keys using Universally Unique Identifiers. UUID key generators only need to have a unique name, they have no other configuration.

UUID-based key generators can be added using the CLI using the following command syntax.

```
/subsystem=cmp/uuid-keygenerator=UNIQUE_NAME:add
```

Example 16.16. Add UUID Key Generator

To add a UUID-based key generator with the name of `uuid_identities`, use this CLI command:

```
/subsystem=cmp/uuid-keygenerator=uuid_identities:add
```

The XML configuration created by this command is:

```
<subsystem xmlns="urn:jboss:domain:cmp:1.0">
    <key-generators>
        <uuid name="uuid_identities" />
    </key-generators>
</subsystem>
```

HiLo Key Generators

HiLo key generators use a database to create and store entity identity keys. HiLo Key generators must have unique names and are configured with properties that specify the datasource used to stored the data as well as the names of the table and columns that store the keys.

HiLo key generators can be added using the CLI using the following command syntax:

```
/subsystem=cmp/hilo-keygenerator=UNIQUE_NAME/:add(property=value,
property=value, ...)
```

Example 16.17. Add a HiLo Key Generator

```
/subsystem=cmp/hilo-keygenerator=DB_KEYS/:add(create-table=false,data-
source=java:jboss/datasources/ExampleDS,drop-table=false,id-
column=cmp_key_ids,select-hi-ddl=select max(cmp_key_ids) from
cmp_key_seq,sequence-column=cmp_key_seq,table-name=cmp-keys)
```

The XML configuration created by this command is:

```
<subsystem xmlns="urn:jboss:domain:cmp:1.0">
  <key-generators>
    <hilo name="DB_KEYS">
      <create-table>false</create-table>
      <data-source>java:jboss/datasources/ExampleDS</data-source>
      <drop-table>false</drop-table>
      <id-column>cmp_key_ids</id-column>
      <select-hi-ddl>select max(cmp_key_ids) from
cmp_key_seq</select-hi-ddl>
      <sequence-column>cmp_key_seq</sequence-column>
      <table-name>cmp-keys</table-name>
    </hilo>
  </key-generators>
</subsystem>
```

[Report a bug](#)

16.9.5. CMP Subsystem Properties for HiLo Key Generators

Table 16.1. CMP Subsystem Properties for HiLo Key Generators

Property	Data type	Description
block-size	long	-
create-table	boolean	If set to TRUE , a table called table-name will be created using the contents of create-table-ddl if that table is not found.
create-table-ddl	string	The DDL commands used to create the table specified in table-name if the table is not found and create-table is set to TRUE .
data-source	token	The data source used to connect to the database.
drop-table	boolean	-
id-column	token	-
select-hi-ddl	string	The SQL command which will return the largest key currently stored.
sequence-column	token	-
sequence-name	token	-
table-name	token	The name of the table used to store the key information.

[Report a bug](#)

Chapter 17. Java Connector Architecture (JCA)

17.1. Introduction

17.1.1. About the Java EE Connector API (JCA)

JBoss Enterprise Application Platform 6 provides full support for the Java EE Connector API (JCA) 1.6 specification. See [JSR 322: Java EE Connector Architecture 1.6](#) for more information about the JCA specification.

A resource adapter is a component that implements the Java EE Connector API architecture. It is similar to a datasource object, however, it provides connectivity from an Enterprise Information System (EIS) to a broader range of heterogeneous systems, such as databases, messaging systems, transaction processing, and Enterprise Resource Planning (ERP) systems.

[Report a bug](#)

17.1.2. Java Connector Architecture (JCA)

The Java EE Connector Architecture (JCA) defines a standard architecture for Java 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.

JCA 1.6 provides features for managing:

- » connections
- » transactions
- » security
- » life-cycle
- » work instances
- » transaction inflow
- » message inflow

JCA 1.6 was developed under the Java Community Process as JSR-322, <http://jcp.org/en/jsr/detail?id=313>.

[Report a bug](#)

17.1.3. Resource Adapters

A resource adapter is a deployable Java EE component that provides communication between a Java EE application and an Enterprise Information System (EIS) using the Java Connector Architecture (JCA) specification. A resource adapter is often provided by EIS vendors to allow easy integration of their products with Java EE applications.

An Enterprise Information Systems 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 Enterprise Application Platform 6. A RAR file may also be included in an Enterprise Archive (EAR) deployment.

[Report a bug](#)

17.2. Configure the Java Connector Architecture (JCA) Subsystem

The JCA subsystem in the JBoss Enterprise Application Platform 6 configuration file controls the general settings for the JCA container and resource adapter deployments.

Key elements of the JCA subsystem

Archive validation

- » This setting whether archive validation will be performed on the deployment units.
- » The following table describes the attributes you can set for archive validation.

Table 17.1. 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 Java EE Connector Architecture 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 (JSR-303) will be performed on the deployment units.
- The following table describes the attributes you can set for bean validation.

Table 17.2. 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 17.3. Work manager attributes

Attribute	Description
name	Specifies the name of the work manager. This is required for custom work managers.
short-running-threads	Thread pool for standard Work instances. Each work manager has one short-running thread pool.
long-running-threads	Thread pool for JCA 1.6 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 17.4. 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 smaller than the maximum thread pool size.
queue-length	The maximum queue length.
max-thread	The maximum thread pool size.
keepalive-time	Specifies the amount of time that pool threads should be kept after doing work.
thread-factory	Reference to the thread factory .

Bootstrap contexts

- » Used to define custom bootstrap contexts.
- » The following table describes the attributes you can set for bootstrap contexts.

Table 17.5. 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

- » 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 17.6. 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.

Procedure 17.1. Configure the JCA subsystem using the Management Console

1. The JCA subsystem of JBoss Enterprise Application Platform 6 can be configured in the Management Console. The JCA configuration options are located in slightly different places in the Management Console depending on how the server is being run.
 - A. If the server is running as a Standalone Server, follow these steps:
 - a. Click on the **Profile** link on the top right to switch to the **Profile** view.
 - b. Ensure the **Profile** section in the navigation panel to the left is expanded.
 - c. Click on **Connector** to expand it, and then click on **JCA**.
 - B. If the server is running as part of a Managed Domain, follow these steps:
 - a. Click on the **Profile** link on the top right to switch to the **Profile** view
 - b. Select the profile you are modifying from the **Profile** menu at the top of the navigation panel on the left.
 - c. Click on **Connector** to expand it, and then click on **JCA**.
2. Configure the settings for the JCA subsystem using the three tabs.
 - a. **Common Config**

The **Common Config** tab contains settings for the cached connection manager, archive validation and bean validation (JSR-303). Each of these is contained in their own tab as well. These settings can be changed by opening the appropriate tab, clicking the edit button, making the required changes, and then clicking on the save button.

Figure 17.1. JCA Common Configuration

b. Work Managers

The **Work Manager** tab 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.

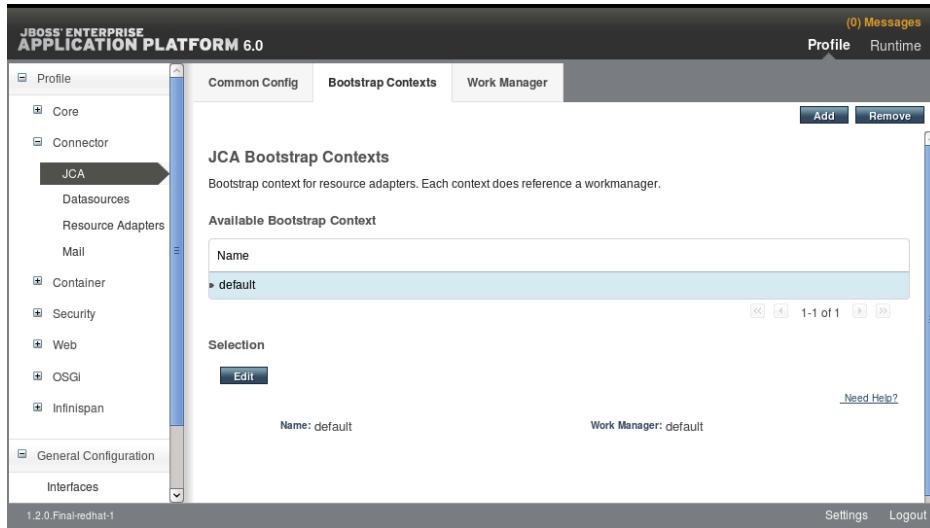
Figure 17.2. Work Managers

The thread pool attributes can be configured here:

Figure 17.3. Work Manager Thread Pools

c. Bootstrap Contexts

The **Bootstrap Contexts** tab 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.

**Figure 17.4. Bootstrap Contexts**

[Report a bug](#)

17.3. Deploy a Resource Adapter

Resource adapters can be deployed to JBoss Enterprise Application Platform 6 using the Management CLI tool, the Web-based Management Console, or by manually copying the files. The process is the same as other deployable artifacts.

Procedure 17.2. Deploy a resource adapter using the Management CLI

1. Open a command prompt for your operating system.

2. Connect to the Management CLI.

A. For Linux, enter the following at the command line:

```
$ EAP_HOME/bin/jboss-cli.sh --connect
$ Connected to standalone controller at localhost:9999
```

B. For Windows, enter the following at a command line:

```
C:>EAP_HOME\bin\jboss-cli.bat --connect
C:> Connected to standalone controller at localhost:9999
```

3. Deploy the resource adapter.

A. To deploy the resource adapter to a standalone server, enter the following at a command line:

```
$ deploy path/to/resource-adapter-name.rar
```

B. To deploy the resource adapter to all server groups in a managed domain, enter the following at a command line:

```
$ deploy path/to/resource-adapter-name.rar --all-server-groups
```

Procedure 17.3. Deploy a resource adapter using the Web-based Management Console

1. Start your JBoss Enterprise Application Platform 6 server.

2. If you have not yet added a management user, add one now. For more information, refer to the

- Getting Started chapter of the Installation guide for JBoss Enterprise Application Platform 6.
3. Open a web browser and navigate to the Management Console. The default location is <http://localhost:9990/console/>. For more information about the Management Console, see [Section 3.2.2, "Log in to the Management Console"](#).
 4. Click on the **Runtime** link on the top right to switch to the Runtime view, then choose **Manage Deployments** in the left navigation panel, and click **Add Content** at the top right.

The screenshot shows the JBoss Enterprise Application Platform 6.0 Management Console. The title bar reads 'JBoss® ENTERPRISE APPLICATION PLATFORM 6.0'. The top navigation bar has tabs for 'Profile' and 'Runtime', with '(0) Messages' and a profile icon. The left sidebar contains links for 'Server', 'Configuration' (with 'Manage Deployments' highlighted), 'Status' (including JVM, Datasources, JPA, Transactions, Web, Webservices), and 'Runtime Operations' (OSGi). The main content area is titled 'Deployments' and shows a table with columns: Name, Runtime Name, Enabled, En/Disable, Update Content, and Remove. A message 'No items!' is displayed below the table. At the bottom of the table are navigation icons for first, previous, next, last, and search. The footer includes '1.3.1.Final-redhat-1', 'Settings', and 'Logout'.

Figure 17.5. Manage Deployments - Add Content

5. Browse to the resource adapter archive and select it. Then click **Next**.

The screenshot shows the 'Upload' step of the deployment selection process. The title bar says 'Upload'. The main content area is titled 'Step1/2: Deployment Selection' and contains the instruction 'Please choose a file that you want to deploy.' Below this is a file input field containing the path '/home/ausertemp/eis.rar' and a 'Browse...' button. At the bottom are 'Next >>' and 'Cancel' buttons.

Figure 17.6. Deployment Selection

6. Verify the deployment names, then click **Save**.

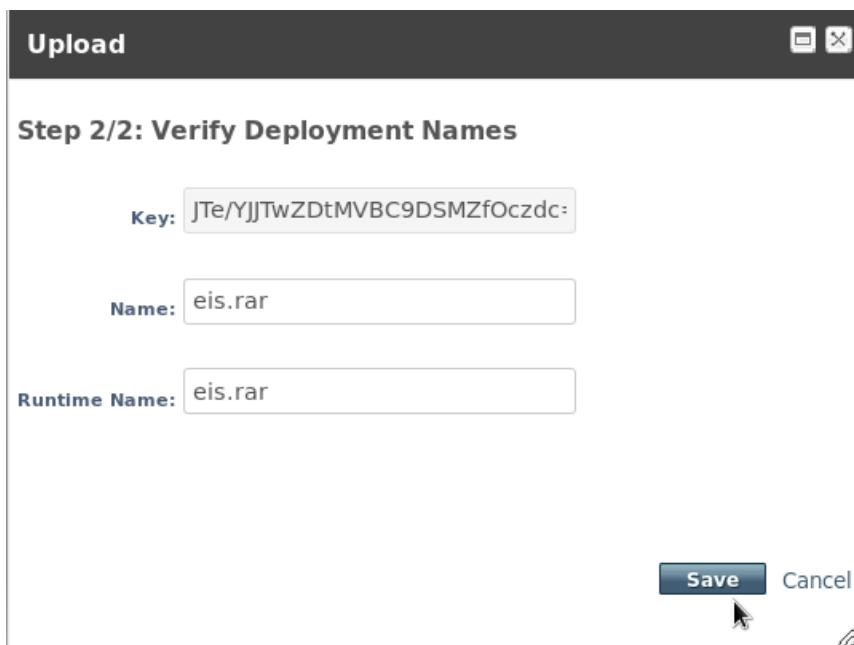


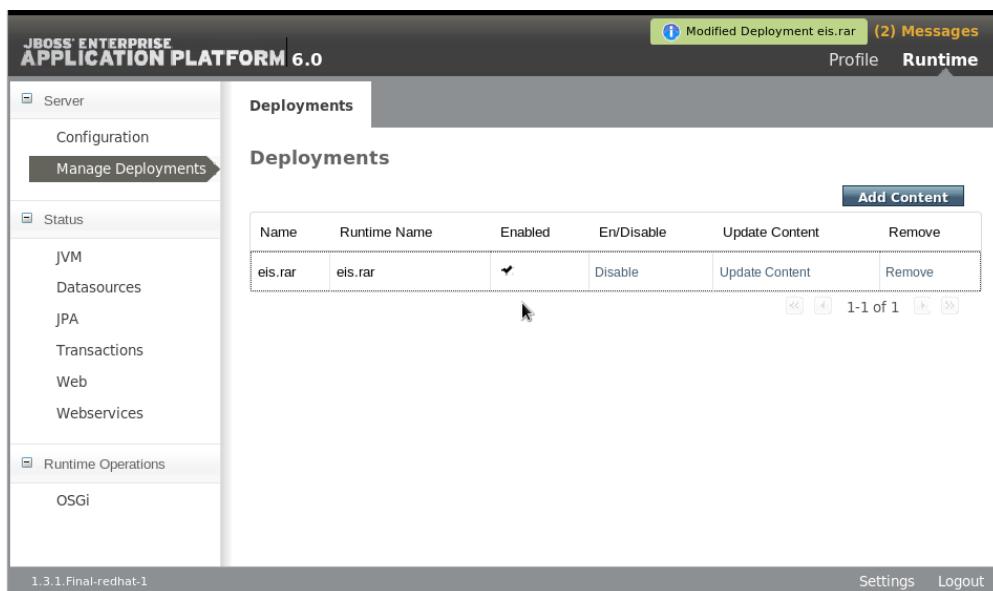
Figure 17.7. Verify Deployment Names

7. The resource adapter archive should now appear in the list in a disabled state. Click the **Enable** link to enable it.

Name	Runtime Name	Enabled	En/Disable	Update Content	Remove
eis.rar	eis.rar	disabled icon	Enable	Update Content	Remove

Figure 17.8. Enable the Deployment

8. A dialog asks "Are you sure?" you want to enable the listed RAR. Click **Confirm**. The resource adapter archive should now appear as **Enabled**.

**Figure 17.9. Deployments****Procedure 17.4. Deploy a resource adapter manually**

- » Copy the resource adapter archive to the server deployments directory,
 - A. For a standalone server, copy the resource adapter archive to the **EAP_HOME/standalone/deployments/** directory.
 - B. For a managed domain, copy the resource adapter archive to the **EAP_HOME/domain/deployments/** directory of the domain controller.

[Report a bug](#)

17.4. Configure a Deployed Resource Adapter

JBoss administrators can configure resource adapters for JBoss Enterprise Application Platform 6 using the Management CLI tool, the Web-based Management Console, or by manually editing the configuration files.

Refer to the vendor document for your resource adapter for information about supported properties and other details.

Procedure 17.5. Configure a resource adapter using the Management CLI

1. Open a command prompt for your operating system.
2. Connect to the Management CLI.
 - A. For Linux, enter the following at the command line:

```
$ EAP_HOME/bin/jboss-cli.sh --connect
$ Connected to standalone controller at localhost:9999
```

- B. For Windows, enter the following at a command line:

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect
C:\> Connected to standalone controller at localhost:9999
```

3. Add the resource adapter configuration.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:add(archive=eis.rar, transaction-support=XATransaction)
{"outcome" => "success"}
```

4. Configure the **server** resource adapter level <config-property>.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/config-properties=server/:add(value=localhost)
>{"outcome" => "success"}
```

5. Configure the **port** resource adapter level <config-property>.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/config-properties=port/:add(value=9000)
>{"outcome" => "success"}
```

6. Add a connection definition for a managed connection factory.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/connection-definitions=cfName:add(class-
name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-
name=java:/eis/AcmeConnectionFactory)
>{"outcome" => "success"}
```

7. Configure the **name** managed connection factory level <config-property>.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/connection-definitions=cfName/config-
properties=name/:add(value=Acme Inc)
>{"outcome" => "success"}
```

8. Add an admin object.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/admin-objects=aoName:add(class-
name=com.acme.eis.ra.EISAdminObjectImpl, jndi-name=java:/eis/AcmeAdminObject)
>{"outcome" => "success"}
```

9. Configure the **threshold** admin object property.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/admin-objects=aoName/config-
properties=threshold/:add(value=10)
>{"outcome" => "success"}
```

10. Activate the resource adapter.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:activate
>{"outcome" => "success"}
```

11. View the newly configured and activated resource adapter.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "archive" => "eis.rar",
        "bean-validation-groups" => undefined,
        "bootstrap-context" => undefined,
        "transaction-support" => "XATransaction",
        "admin-objects" => {"aoName" => {
            "class-name" => "com.acme.eis.ra.EISAdminObjectImpl",
            "enabled" => true,
            "jndi-name" => "java:/eis/AcmeAdminObject",
            "use-java-context" => true,
            "config-properties" => {"threshold" => {"value" => 10}}
        }},
        "config-properties" => {
            "server" => {"value" => "localhost"},
            "port" => {"value" => 9000}
        },
        "connection-definitions" => {"cfName" => {
            "allocation-retry" => undefined,
            "allocation-retry-wait-millis" => undefined,
            "background-validation" => false,
            "background-validation-millis" => undefined,
            "blocking-timeout-wait-millis" => undefined,
            "class-name" => "com.acme.eis.ra.EISManagedConnectionFactory",
            "enabled" => true,
            "flush-strategy" => "FailingConnectionOnly",
            "idle-timeout-minutes" => undefined,
            "interleaving" => false,
            "jndi-name" => "java:/eis/AcmeConnectionFactory",
            "max-pool-size" => 20,
            "min-pool-size" => 0,
            "no-recovery" => undefined,
            "no-tx-separate-pool" => false,
            "pad-xid" => false,
            "pool-prefill" => false,
            "pool-use-strict-min" => false,
            "recovery-password" => undefined,
            "recovery-plugin-class-name" => undefined,
            "recovery-plugin-properties" => undefined,
            "recovery-security-domain" => undefined,
            "recovery-username" => undefined,
            "same-rm-override" => undefined,
            "security-application" => undefined,
            "security-domain" => undefined,
            "security-domain-and-application" => undefined,
            "use-ccm" => true,
            "use-fast-fail" => false,
            "use-java-context" => true,
            "use-try-lock" => undefined,
            "wrap-xa-resource" => true,
            "xa-resource-timeout" => undefined,
            "config-properties" => {"name" => {"value" => "Acme Inc"}}
        }}
    }
}
```

Procedure 17.6. Configure a resource adapter using the Web-based Management Console

1. Start your JBoss Enterprise Application Platform 6 server.
2. If you have not yet added a management user, add one now. For more information, refer to the Getting Started chapter of the Installation guide for JBoss Enterprise Application Platform 6.
3. Open a web browser and navigate to the Management Console. The default location is <http://localhost:9990/console/>. For more information about the Management Console, see [Section 3.2.2, “Log in to the Management Console”](#).
4. Click on the **Profile** link on the top right to switch to the Profile view. Choose **Resource Adapters** in the left navigation panel, then click **Add** at the top right.

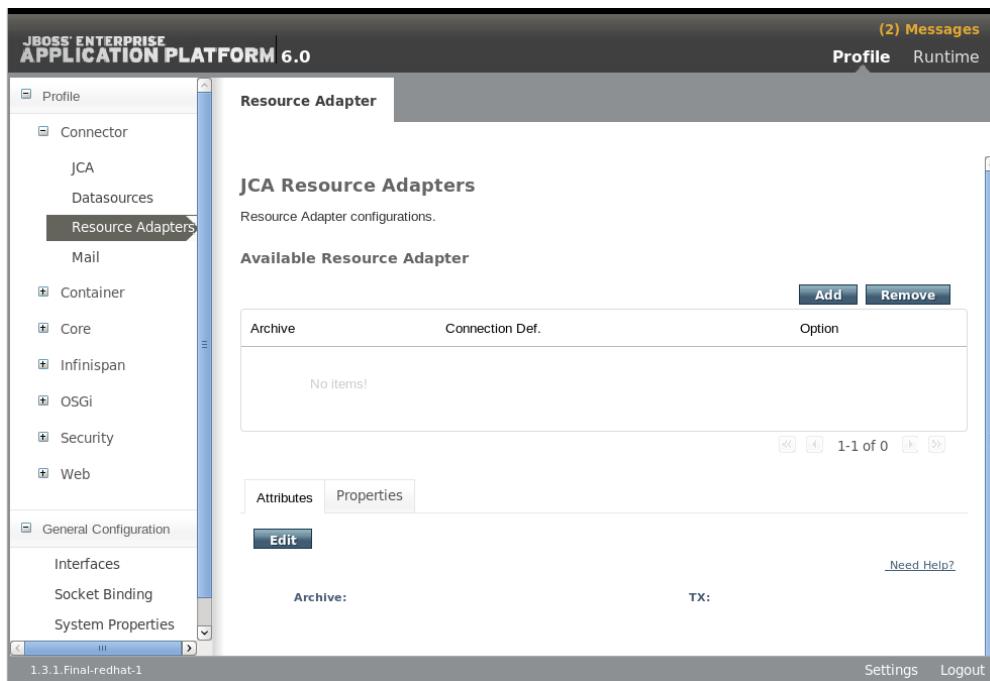


Figure 17.10. JCA Resource Adapters

- Enter the archive name and choose transaction type **XATransaction** from the TX: drop-down box. Then click **Save**.

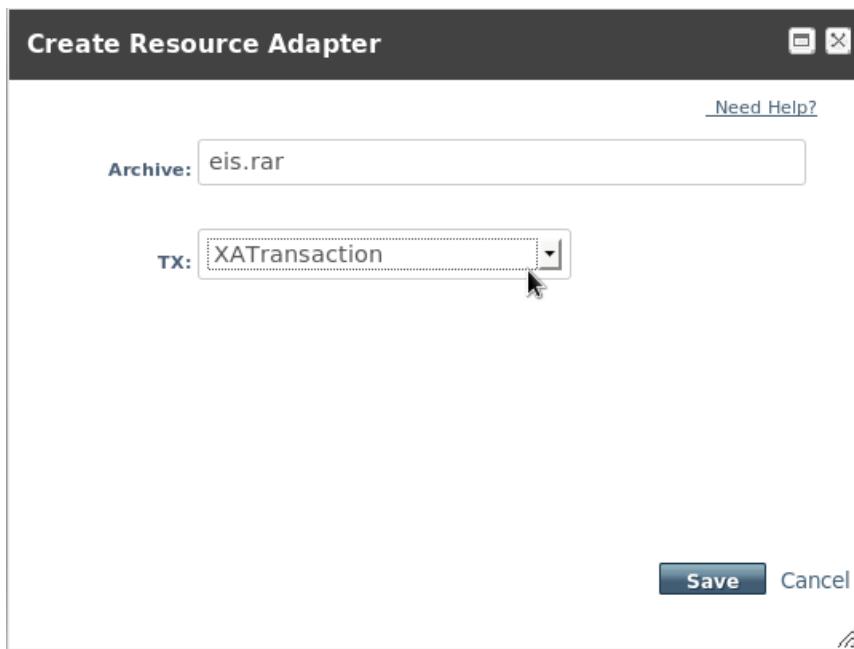


Figure 17.11. Create Resource Adapter

- Select the **Properties** tab, then click **Add** to add resource adapter properties.

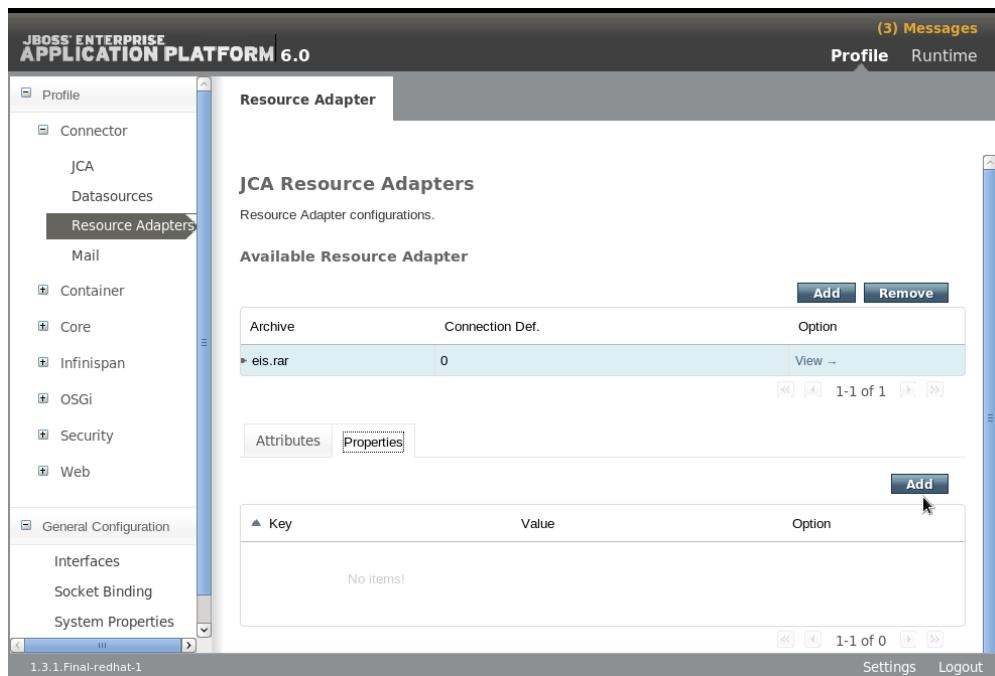


Figure 17.12. Add Resource Adapter Properties

7. Enter **server** for the **Name** and the host name, for example **localhost**, for the **Value**. Then click **Save** to save the property.

Name:	server
Value:	localhost

Save **Cancel**

Figure 17.13. Add Resource Adapter Server Property

8. Enter **port** for the **Name** and the port number, for example **9000**, for the **Value**. Then click **Save** to save the property.

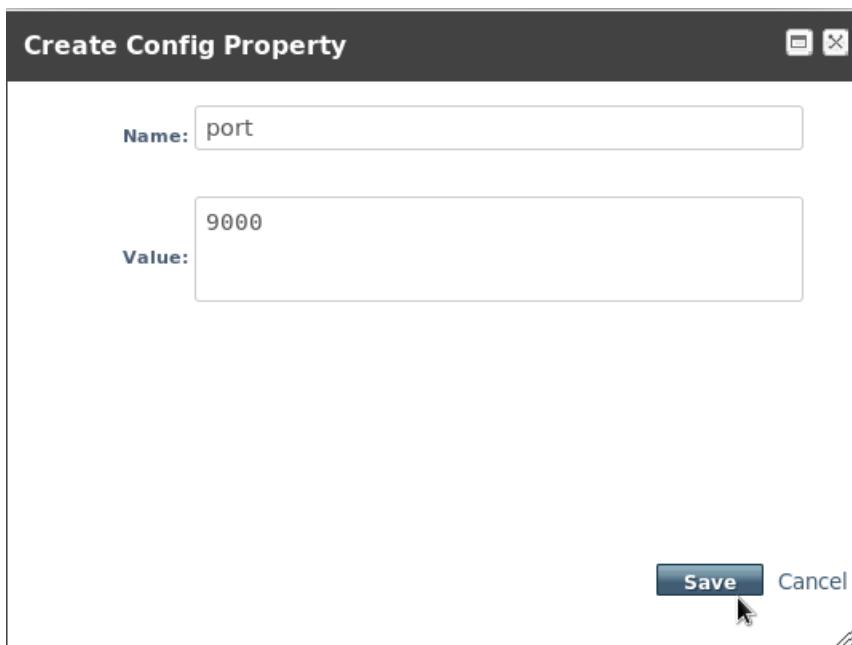


Figure 17.14. Add Resource Adapter Port Property

9. The **server** and **port** properties now appear in the **Properties** panel. Click the **View** link under the **Option** column for the listed resource adapter to view the **Connection Definitions**.

Key	Value	Option
server	localhost	Remove
port	9000	Remove

Figure 17.15. Resource Adapter Server Properties Complete

10. Click **Add** in the upper right side of the page to add a connection definition.

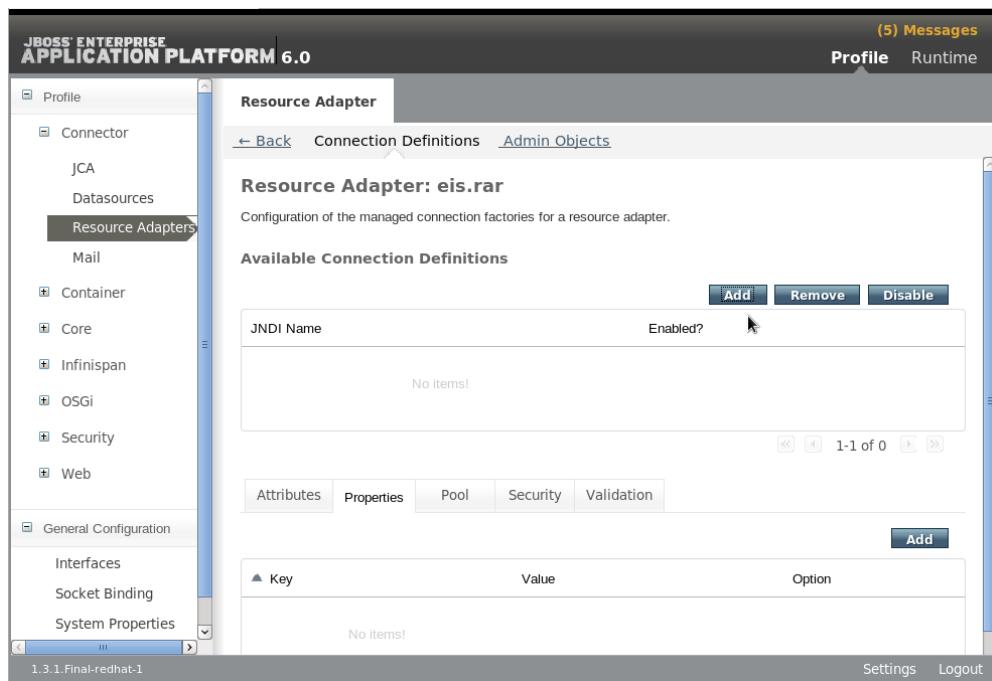


Figure 17.16. Add a Connection Definition

11. Enter the **JNDI Name** and the fully qualified class name of the **Connection Class**. Then click **Next**.

The screenshot shows the "Create Connection Definition" dialog. The title bar says "Create Connection Definition". The main area is titled "Connection Definition Step1/2". It contains two input fields: "JNDI Name:" with the value "java:/eis/AcmeConnectionFactory" and "Connection Class:" with the value "com.acme.eis.ra.EISManagedConnectionFactory". There is a "Need Help?" link above the fields. At the bottom are "Next >" and "Cancel" buttons, with "Next >" being the active button.

Figure 17.17. Create Connection Definition Property - Step 1

12. Click **Add** to enter the **Key** and **Value** data for this connection definition.

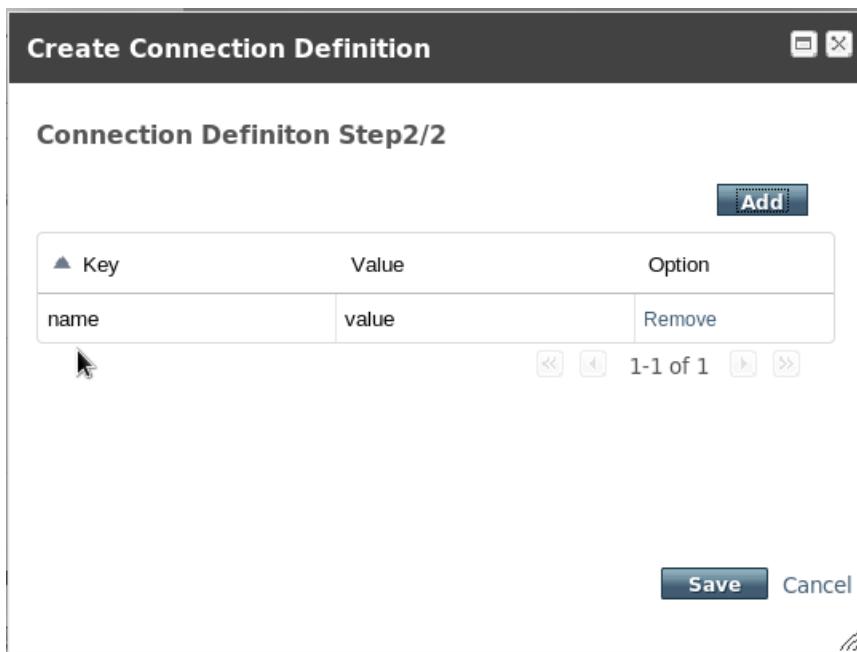


Figure 17.18. Create Connection Definition Property - Step 2

- Click in the **name** field under the **Key** column to enable data entry for the field. Type the property name and press enter when done. Click in the **value** field under the **Value** column to enable data entry for the field. Enter the value for the property and press enter when done. Then click **Save** to save the property.

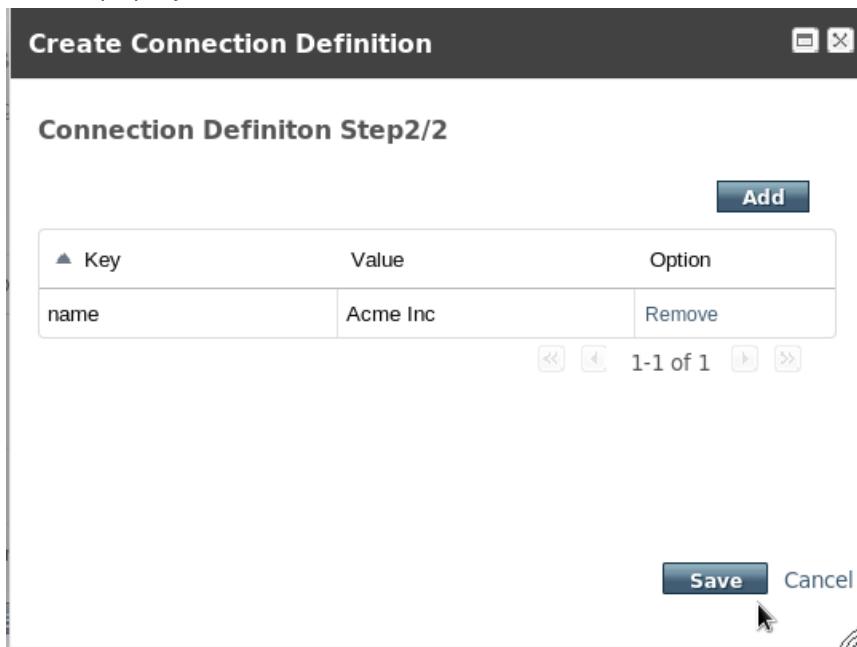


Figure 17.19. Create Connection Definition Property - Step 2

- The connection definition is complete, but disabled. Click **Enable** to enable the connection definition.

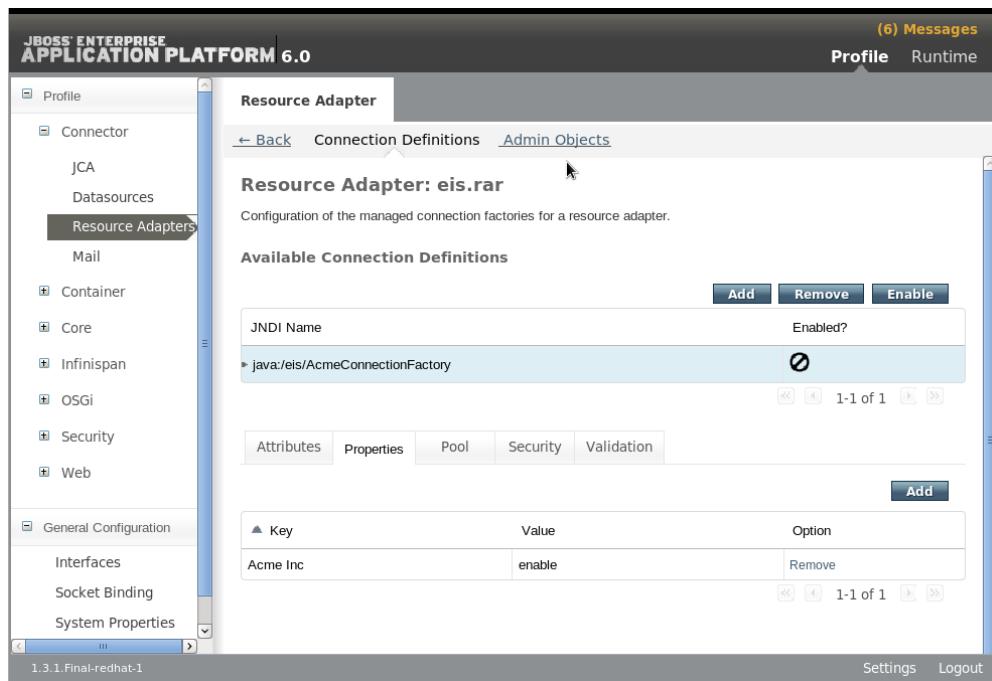


Figure 17.20. Create Connection Definition - Disabled

15. A dialog asks "Really modify Connection Definition?" for the JNDI name. Click **Confirm**. The connection definition should now appear as **Enabled**.

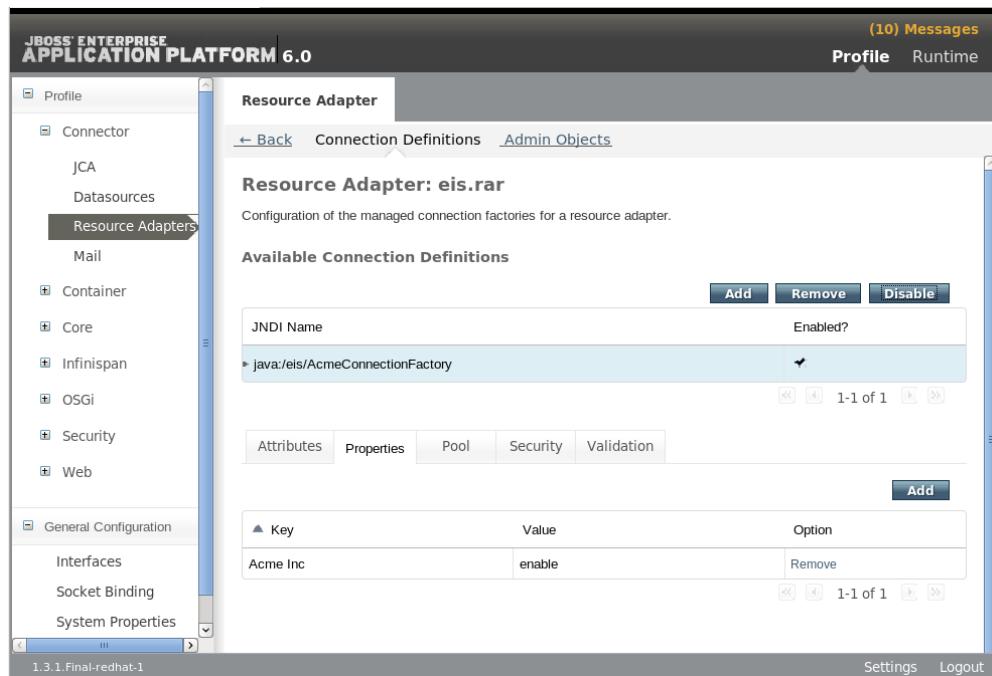


Figure 17.21. Connection Definition Enabled

16. Click the **Admin Objects** tab in top middle of the page to create and configure admin objects. Then click **Add**.

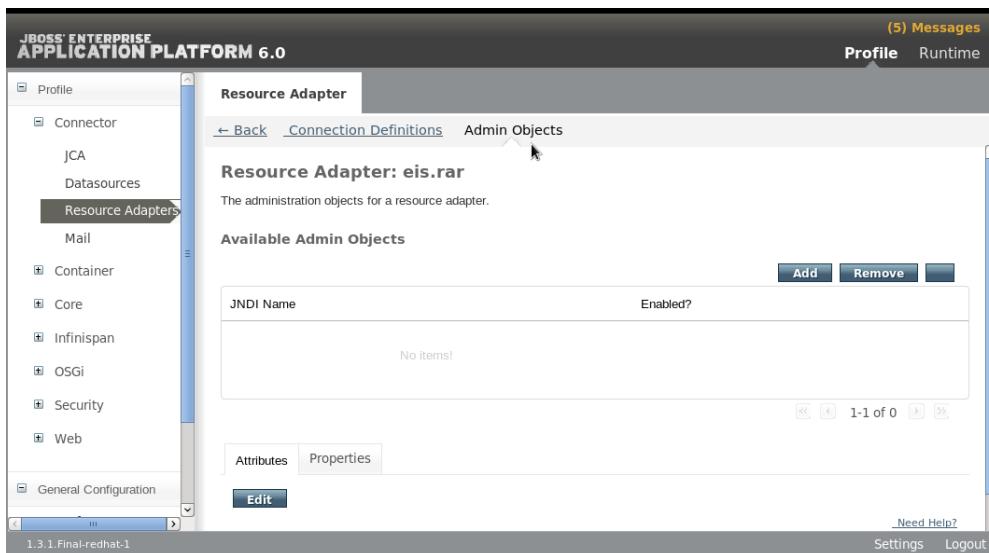


Figure 17.22. Available Admin Objects

17. Enter the **JNDI Name** and the fully qualified **Class Name** for the admin object. Then click **Save**.

The screenshot shows a 'Create admin object' dialog box. It has fields for 'JNDI Name' (containing 'java:/eis/AcmeAdminObject') and 'Class Name' (containing 'com.acme.eis.ra.EISAdminObjectImpl'). There is a 'Need Help?' link. At the bottom are 'Save' and 'Cancel' buttons, with 'Save' being highlighted.

Figure 17.23. Create Admin Object

18. Select the **Properties** tab, then click **Add** to add admin object properties.

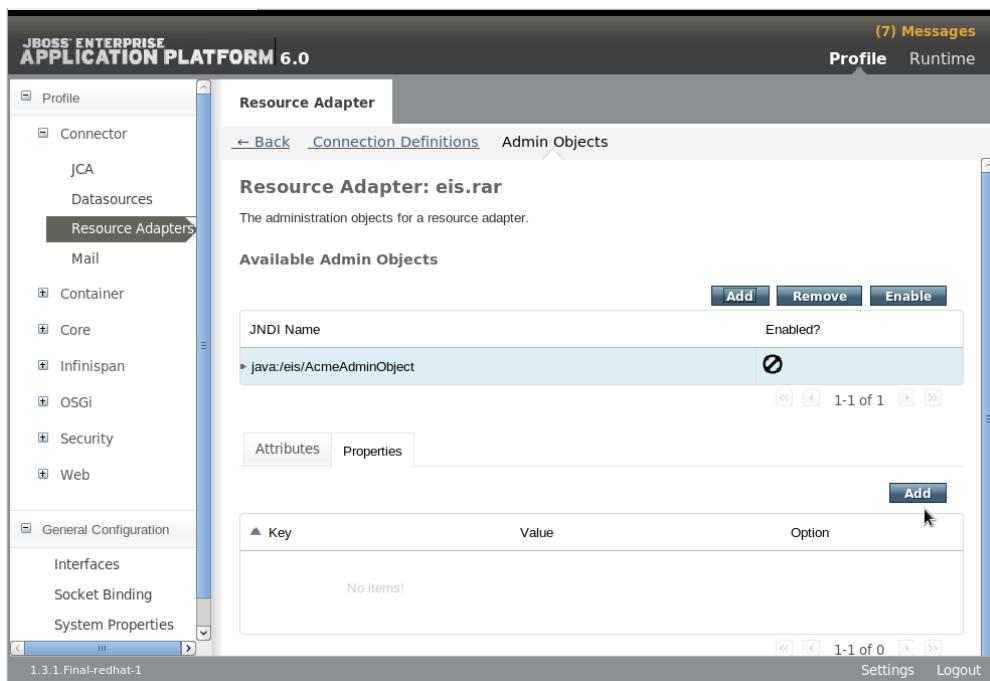


Figure 17.24. Add Admin Object Properties

19. Enter an admin object configuration property, for example **threshold**, in the **Name** field. Enter the configuration property value, for example **10**, in the **Value** field. Then click **Save** to save the property.

The dialog box is titled 'Create Config Property'. It has two input fields: 'Name:' containing 'threshold' and 'Value:' containing '10'. At the bottom are 'Save' and 'Cancel' buttons, with 'Save' being the active button indicated by a cursor icon.

Figure 17.25. Create Admin Object Configuration Property

20. The admin object is complete, but disabled. Click **Enable** to enable the admin object.

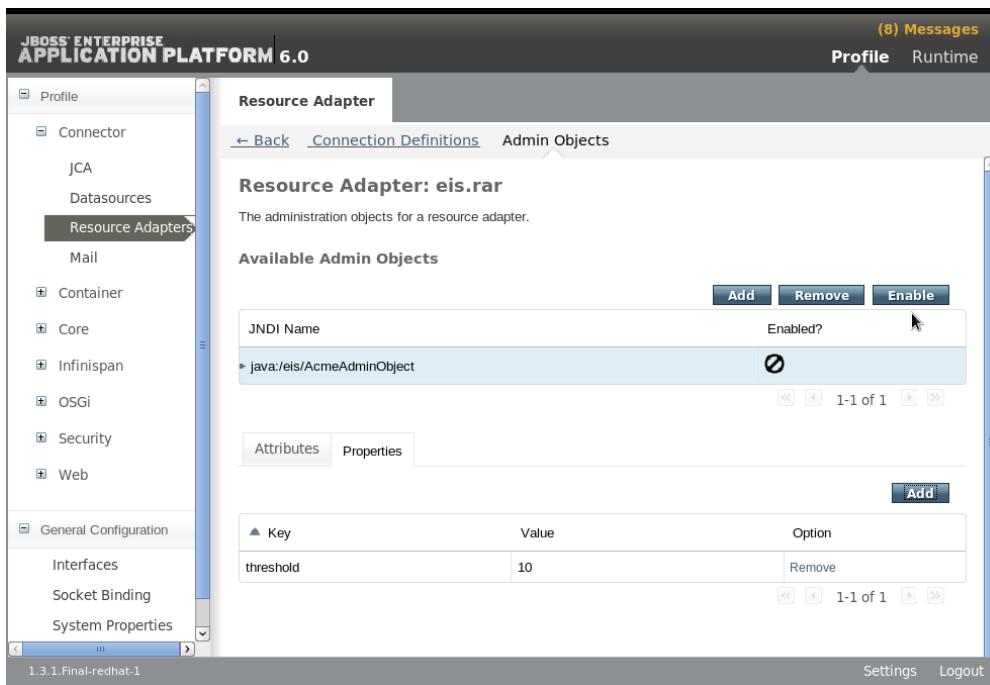


Figure 17.26. Admin Object - Disabled

21. A dialog asks "Really modify Admin Object?" for the JNDI name. Click **Confirm**. The admin object should now appear as **Enabled**.

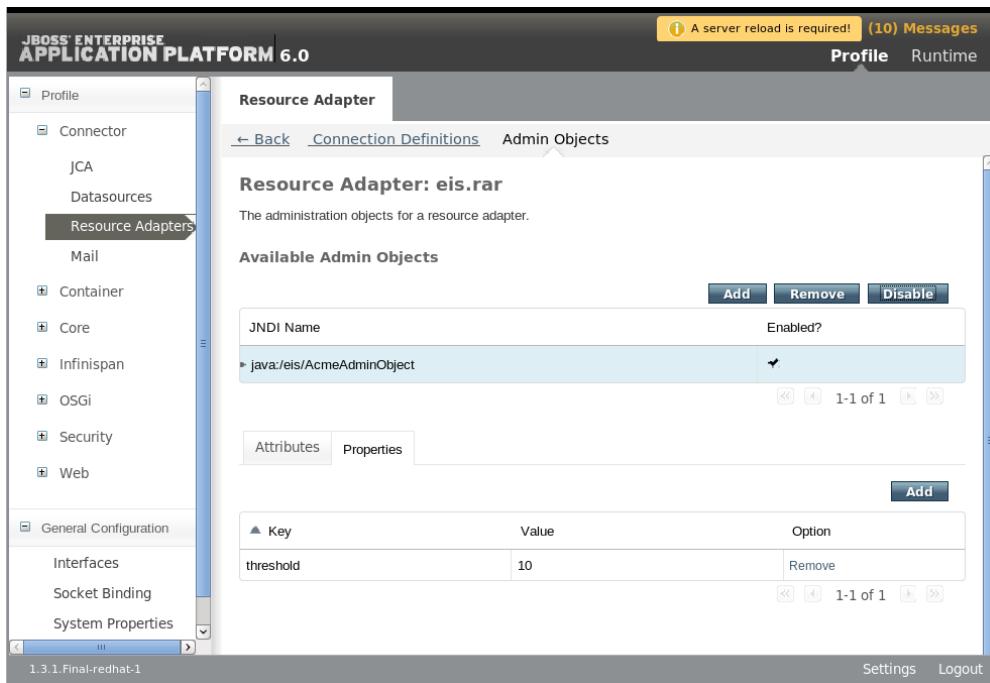
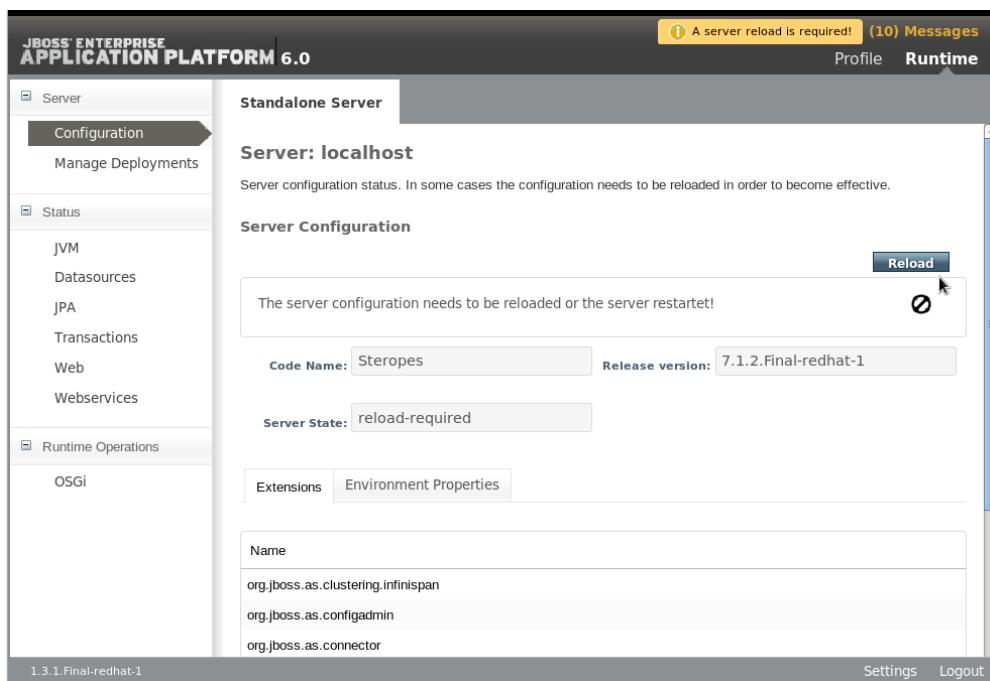
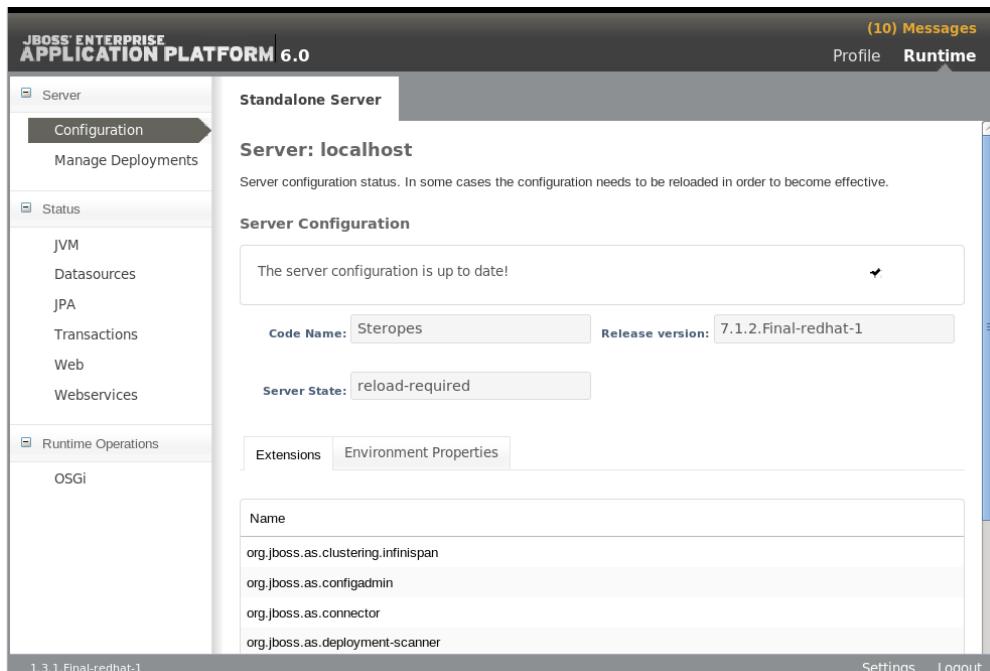


Figure 17.27. Connection Definition Enabled

22. You must reload the server configuration to complete the process. Click on the **Runtime** link on the top right to switch to the Runtime view, then choose **Configuration** in the left navigation panel, and click **Reload** at the right.

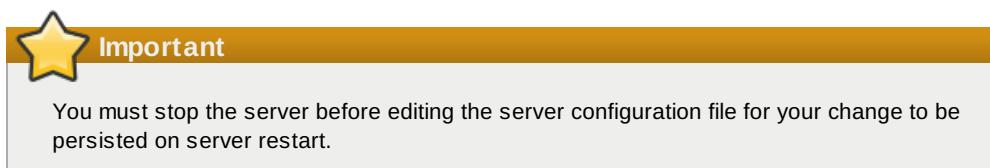
**Figure 17.28. Server Configuration Reload**

23. A dialog asks "Do you want to reload the server configuration?" for the specified server. Click **Confirm**. The server configuration is up to date.

**Figure 17.29. Connection Definition Enabled**

Procedure 17.7. Configure a resource adapter manually

1. Stop the JBoss Enterprise Application Platform server.



2. Open the server configuration file for editing.
 - A. For a standalone server, this is the `EAP_HOME/standalone/configuration/standalone.xml` file.
 - B. For a managed domain, this is the `EAP_HOME/domain/configuration/domain.xml` file.
3. Find the `urn:jboss:domain:resource-adapters` subsystem in the configuration file.
4. If there are no resource adapters defined for this subsystem, first replace:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.0"/>
```

with this:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.0">
  <resource-adapters>
    <!-- <resource-adapter> configuration listed below -->
  </resource-adapters>
</subsystem>
```

5. Replace the `<!-- <resource-adapter> configuration listed below -->` with the XML definition for your resource adapter. The following is the XML representation of the resource adapter configuration created using the Management CLI and Web-based Management Console described above.

```
<resource-adapter>
  <archive>
    eis.rar
  </archive>
  <transaction-support>XATransaction</transaction-support>
  <config-property name="server">
    localhost
  </config-property>
  <config-property name="port">
    9000
  </config-property>
  <connection-definitions>
    <connection-definition class-
name="com.acme.eis.ra.EISManagedConnectionFactory"
      jndi-name="java:/eis/AcmeConnectionFactory"
      pool-name="java:/eis/AcmeConnectionFactory">
      <config-property name="name">
        Acme Inc
      </config-property>
    </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.acme.eis.ra.EISAdminObjectImpl"
      jndi-name="java:/eis/AcmeAdminObject"
      pool-name="java:/eis/AcmeAdminObject">
      <config-property name="threshold">
        10
      </config-property>
    </admin-object>
  </admin-objects>
</resource-adapter>
```

6. Start the server

Relaunch the JBoss Enterprise Application Platform server to start it running with the new configuration.

[Report a bug](#)

17.5. Resource Adapter Descriptor Reference

The following tables describe the resource adapter descriptor elements.

Table 17.7. Main elements

Element	Description
bean-validation-groups	Specifies bean validation group that should be used
bootstrap-context	Specifies the unique name of the bootstrap context that should be used
config-property	The config-property specifies resource adapter configuration properties.
transaction-support	Define the type of transaction supported by this resource adapter. Valid values are: NoTransaction, LocalTransaction, XATransaction
connection-definitions	Specifies the connection definitions
admin-objects	Specifies the administration objects

Table 17.8. Bean validation groups elements

Element	Description
bean-validation-group	Specifies the fully qualified class name for a bean validation group that should be used for validation

Table 17.9. Connection definition / admin object attributes

Attribute	Description
class-name	Specifies the fully qualified class name of a managed connection factory or admin object
jndi-name	Specifies the JNDI name
enabled	Should the object in question be activated
use-java-context	Specifies if a java:/ JNDI context should be used
pool-name	Specifies the pool name for the object
use-ccm	Enable the cache connection manager

Table 17.10. Connection definition elements

Element	Description
config-property	The config-property specifies managed connection factory configuration properties.
pool	Specifies pooling settings
xa-pool	Specifies XA pooling settings
security	Specifies security settings
timeout	Specifies time out settings
validation	Specifies validation settings
recovery	Specifies the XA recovery settings

Table 17.11. Pool elements

Element	Description
min-pool-size	The min-pool-size element indicates the minimum number of connections a pool should hold. These are not created until a Subject is known from a request for a connection. This default to 0
max-pool-size	The max-pool-size element indicates the maximum number of connections for a pool. No more than max-pool-size connections will be created in each sub-pool. This defaults to 20.
prefill	Whether to attempt to prefill the connection pool. Default is false
use-strict-min	Specifies if the min-pool-size should be considered strictly. Default false
flush-strategy	Specifies how the pool should be flush in case of an error. Valid values are: FailingConnectionOnly (default), IdleConnections, EntirePool

Table 17.12. XA pool elements

Element	Description
min-pool-size	The min-pool-size element indicates the minimum number of connections a pool should hold. These are not created until a Subject is known from a request for a connection. This default to 0
max-pool-size	The max-pool-size element indicates the maximum number of connections for a pool. No more than max-pool-size connections will be created in each sub-pool. This defaults to 20.
prefill	Whether to attempt to prefill the connection pool. Default is false
use-strict-min	Specifies if the min-pool-size should be considered strictly. Default false
flush-strategy	Specifies how the pool should be flush in case of an error. Valid values are: FailingConnectionOnly (default), IdleConnections, EntirePool
is-same-rm-override	The is-same-rm-override element allows one to unconditionally set whether the javax.transaction.xa.XAResource.isSameRM(XAResource) returns true or false
interleaving	An element to enable interleaving for XA connection factories
no-tx-separate-pools	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	Should the Xid be padded
wrap-xa-resource	Should the XAResource instances be wrapped in a org.jboss.tm.XAResourceWrapper instance

Table 17.13. Security elements

Element	Description
application	Indicates that application supplied parameters (such as from getConnection(user, pw)) are used to distinguish connections in the pool.
security-domain	Indicates Subject (from security domain) are used to distinguish connections in the pool. The content of the security-domain is the name of the JAAS security manager that will handle authentication. This name correlates to the JAAS login-config.xml descriptor application-policy/name attribute.
security-domain-and-application	Indicates that either application supplied parameters (such as from getConnection(user, pw)) or Subject (from security domain) are used to distinguish connections in the pool. The content of the security-domain is the name of the JAAS security manager that will handle authentication. This name correlates to the JAAS login-config.xml descriptor application-policy/name attribute.

Table 17.14. Time out elements

Element	Description
blocking-timeout-millis	The blocking-timeout-millis element indicates the maximum time in milliseconds to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for a permit for a connection, and will never throw an exception if creating a new connection takes an inordinately long time. The default is 30000 (30 seconds).
idle-timeout-minutes	The idle-timeout-minutes elements indicates 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 1/2 the smallest idle-timeout-minutes of any pool.
allocation-retry	The allocation retry element indicates the number of times that allocating a connection should be tried before throwing an exception. The default is 0.
allocation-retry-wait-millis	The allocation retry wait millis element indicates the time in milliseconds to wait between retrying to allocate a connection. The default is 5000 (5 seconds).
xa-resource-timeout	Passed to XAResource.setTransactionTimeout(). Default is zero which does not invoke the setter. Specified in seconds

Table 17.15. Validation elements

Element	Description
background-validation	An element to specify that connections should be validated on a background thread versus being validated prior to use
background-validation-minutes	The background-validation-minutes element specifies the amount of time, in minutes, that background validation will run.
use-fast-fail	Whether fail a connection allocation on the first connection if it is invalid (true) or keep trying until the pool is exhausted of all potential connections (false). Default is false

Table 17.16. Admin object elements

Element	Description
config-property	Specifies an administration object configuration property.

Table 17.17. Recovery elements

Element	Description
recover-credential	Specifies the user name / password pair or security domain that should be used for recovery.
recover-plugin	Specifies an implementation of the org.jboss.jca.core.spi.recovery.RecoveryPlugin class.

The deployment schemas are defined in **jboss-as-resource-adapters_1_0.xsd** and http://docs.jboss.org/ironjacamar/schema/ironjacamar_1_0.xsd for automatic activation.

[Report a bug](#)

17.6. Deploy the WebSphere MQ Resource Adapter

About WebSphere MQ

WebSphere MQ is IBM's Messaging Oriented Middleware (MOM) software that allows applications on distributed systems to communicate with each other. This is accomplished through the use of messages and message queues. WebSphere MQ is responsible for delivering messages to the message queues and for transferring data to other queue managers using message channels. For more information about WebSphere MQ, see [WebSphere MQ](#).

Summary

This topic covers the steps to deploy and configure the WebSphere MQ Resource Adapter in JBoss Enterprise Application Platform 6. This can be accomplished by manually editing configuration files, using the Management CLI tool, or using the web-based Management Console.

Prerequisites

Before you get started, you must verify the version of the WebSphere MQ resource adapter and understand some of the WebSphere MQ configuration properties.

- ▶ The WebSphere MQ resource adapter is supplied as a Resource Archive (RAR) file called **wmq.jmsra-VERSION.rar**. You must use version **7.0.1.7** or later.
- ▶ You must know the values of the following WebSphere MQ configuration properties. Refer to the WebSphere MQ product documentation for details about these properties.
 - MQ.QUEUE.MANAGER: The name of the WebSphere MQ queue manager
 - MQ.HOST.NAME: The host name used to connect to the WebSphere MQ queue manager
 - MQ.CHANNEL.NAME: The server channel used to connect to the WebSphere MQ queue manager
 - MQ.QUEUE.NAME: The name of the destination queue
 - MQ.PORT: The port used to connect to the WebSphere MQ queue manager
 - MQ.CLIENT: The transport type
- ▶ For outbound connections, you must also be familiar with the following configuration property:
 - MQ.CONNECTIONFACTORY.NAME: The name of the connection factory instance that will provide the connection to the remote system

Note

In order to support transactions with the WebSphereMQ resource adapter, you must do the following:

- ▶ Repackage the **wmq.jmsra-VERSION.rar** archive to include the **mqetclient.jar**. You can use the following command - be sure to replace the **VERSION** the correct version number:
`jar -uf wmq.jmsra-VERSION.rar mqetclient.jar`
- ▶ Change the **<transaction-support>** element to value to **XATransaction**.

Procedure 17.8. Deploy the Resource Adapter Manually

1. Copy the **wmq.jmsra-VERSION.rar** file to the **EAP_HOME/standalone/deployments/** directory.
2. Add the resource adapter to the server configuration file.

- a. Open the **EAP_HOME/standalone/configuration/standalone-full.xml** file in an editor.
- b. Find the **urn:jboss:domain:resource-adapters** subsystem in the configuration file.
- c. If there are no resource adapters defined for this subsystem, first replace:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.0"/>
```

with this:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.0">
<resource-adapters>
    <!-- <resource-adapter> configuration listed below -->
</resource-adapters>
</subsystem>
```

- d. Replace the **<!-- <resource-adapter> configuration listed below -->** with the following:

```
<resource-adapter>
<archive>
    wmq.jmsra-VERSION.rar
</archive>
<transaction-support>NoTransaction</transaction-support>
<connection-definitions>
    <connection-definition class-
        name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
        jndi-
        name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
        pool-name="MQ.CONNECTIONFACTORY.NAME">
        <config-property name="channel">
            MQ.CHANNEL.NAME
        </config-property>
        <config-property name="transportType">
            MQ.CLIENT
        </config-property>
        <config-property name="queueManager">
            MQ.QUEUE.MANAGER
        </config-property>
    </connection-definition>
</connection-definitions>
<admin-objects>
    <admin-object
        class-
        name="com.ibm.mq.connector.outbound.MQQueueProxy"
        jndi-name="java:jboss/MQ.QUEUE.NAME"
        pool-name="MQ.QUEUE.NAME">
        <config-property name="baseQueueName">
            MQ.QUEUE.NAME
        </config-property>
    </admin-object>
</admin-objects>
</resource-adapter>
```

Be sure to replace the **VERSION** with the actual version in the name of the RAR.

- e. If you want to change the default provider for the EJB3 messaging system in JBoss Enterprise Application Platform 6 from HornetQ to WebSphere MQ, modify the **urn:jboss:domain:ejb3:1.2** subsystem as follows:

Replace:

```
<mdb>
  <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

with:

```
<mdb>
  <resource-adapter-ref resource-adapter-name="wmq.jmsra-
VERSION.rar"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

Be sure to replace the **VERSION** with the actual version in the name of the RAR.

Procedure 17.9. Modify the MDB code to use the resource adapter

- Configure the ActivationConfigProperty and ResourceAdapter in the MDB code as follows:

```
@MessageDriven( name="WebSphereMQMDB",
  activationConfig =
  {
    @ActivationConfigProperty(propertyName =
"destinationType", propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "useJNDI", propertyValue =
>false"),
    @ActivationConfigProperty(propertyName = "hostName", propertyValue =
"MQ.HOST.NAME"),
    @ActivationConfigProperty(propertyName = "port", propertyValue =
"MQ.PORT"),
    @ActivationConfigProperty(propertyName = "channel", propertyValue =
"MQ.CHANNEL.NAME"),
    @ActivationConfigProperty(propertyName = "queueManager", propertyValue =
"MQ.QUEUE.MANAGER"),
    @ActivationConfigProperty(propertyName = "destination", propertyValue =
"MQ.QUEUE.NAME"),
    @ActivationConfigProperty(propertyName = "transportType", propertyValue
= "MQ.CLIENT")
  })
  @ResourceAdapter(value = "wmq.jmsra-VERSION.rar")
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public class WebSphereMQMDB implements MessageListener { }
```

Be sure to replace the **VERSION** with the actual version in the name of the RAR.

[Report a bug](#)

Chapter 18. Deploy JBoss Enterprise Application Platform 6 on Amazon EC2

18.1. Introduction

18.1.1. About Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a service operated by amazon.com that provides customers with a customizable virtual computing environment. An Amazon Machine Image (AMI) can be booted using the service to create a virtual machine or instance. Users can install whatever software they require on an instance and are charged according to the capacity used. Amazon EC2 is designed to be flexible and allow users to quickly scale their deployed applications.

You can read more about it at the Amazon EC2 website, <http://aws.amazon.com/ec2/>.

[Report a bug](#)

18.1.2. About Amazon Machine Instances (AMIs)

An Amazon Machine Image (AMI) is a template for a EC2 virtual machine instance. Users create EC2 instances by selecting an appropriate AMI to create the instance from. The primary component of an AMI is a read-only filesystem that contains an installed operating system as well as other software. Each AMI has different software installed for different use cases. Amazon EC2 includes many AMIs to choose from provided by both amazon.com and third parties. Users can also create their own custom AMIs.

[Report a bug](#)

18.1.3. About JBoss Cloud Access

JBoss Cloud Access is a Red Hat subscription feature that provides support for JBoss Enterprise Application Platform 6 on Red Hat certified cloud infrastructure providers such as Amazon EC2. JBoss Cloud Access allows you to move your subscriptions between traditional servers and public cloud-based resources in a simple and cost-effective manner.

You can find out more details about it at <http://www.redhat.com/solutions/cloud/access/jboss/>.

[Report a bug](#)

18.1.4. JBoss Cloud Access Features

Membership in the JBoss Cloud Access program provides access to supported private Amazon Machine Images (AMIs) created by Red Hat.

The Red Hat AMIs have the following software pre-installed and fully supported by Red hat:

- ▶ Red Hat Enterprise Linux 6
- ▶ JBoss Enterprise Application Platform 6
- ▶ The JBoss Operations Network (JON) 3 agent
- ▶ Product updates with RPMs using Red Hat Update Infrastructure.

Each of the Red Hat AMIs are only a starting point, requiring further configuration to the requirements of your application.



Important

JBoss Cloud Access does not currently provide support for the full-ha profile, in either standalone instances or a managed domain.

[Report a bug](#)

18.1.5. Supported Amazon EC2 Instance Types

JBoss Cloud Access supports the following Amazon EC2 instance types. Refer to the *Amazon EC2 User Guide* for more details about each instance type,
<http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/instance-types.html>.

Table 18.1. Supported Amazon EC2 Instance Types

Instance Type	Description
Standard Instance	Standard Instances are general purpose environments with a balanced memory-to-CPU ratio.
High Memory Instance	High Memory Instances have more memory allocated to them than Standard Instances. High Memory Instances are suited for high throughput applications such as databases or memory caching applications.
High CPU Instance	High CPU Instances have more CPU resources allocated than memory and are suited for relatively low throughput but CPU intensive applications.

**Important**

The instance type **Micro (t1.micro)** is not suitable for deployment of JBoss Enterprise Application Platform.

[Report a bug](#)

18.1.6. Supported Red Hat AMIs

The supported Red Hat AMIs can be identified by their AMI Name.

The JBoss Enterprise Application Platform 6 AMIs are named using the following syntax:

RHEL-osversion-JBEAP-6.0.0-arch-creationdate

osversion is the version number of Red Hat Enterprise Linux installed in the AMI. Example **6.2**.

arch is the architecture of the AMI. This will be **x86_64** or **i386**.

creationdate is the date that the AMI was created in the format of **YYYYMMDD**. Example **20120501**.

Example AMI name: **RHEL-6.2-JBEAP-6.0.0-x86_64-20120501**.

[Report a bug](#)

18.2. Deploying JBoss Enterprise Application Platform 6 on Amazon EC2**18.2.1. Overview of Deploying JBoss Enterprise Application Platform 6 on Amazon EC2**

JBoss Enterprise Application Platform 6 can be deployed using the Amazon EC2 AMI. The AMI contains everything that is required for deployment of clustered and non-clustered instances.

Deploying a non-clustered instances is the easiest scenario. It requires only that you make a few configuration changes to specify your application deployment when creating the instance.

Deploying clustered instances is more complicated. In addition to your clustered instances you are required to deploy a JBoss Enterprise Application Platform 6 instance to act as a mod_cluster proxy and an S3 bucket for the S3_PING JGroups discovery protocol. Red Hat also recommends the creation of a Virtual Private Cloud to contain your cluster.

Each of these steps is detailed below but it is assumed that you have some experience with JBoss Enterprise Application Platform 6, Red Hat Enterprise Linux 6 and Amazon EC2.

The following documentation is recommended additional reference:

- ▶ JBoss Enterprise Application Platform 6, https://access.redhat.com/knowledge/docs/JBoss_Enterprise_Application_Platform/.
- ▶ Red Hat Enterprise Linux 6, https://access.redhat.com/knowledge/docs/Red_Hat_Enterprise_Linux/.
- ▶ Amazon Web Services, <http://aws.amazon.com/documentation/>.

[Report a bug](#)

18.2.2. Non-clustered JBoss Enterprise Application Platform 6

18.2.2.1. About Non-clustered Instances

A non-clustered instance is a single Amazon EC2 instance running JBoss Enterprise Application Platform 6. It is not part of a cluster.

[Report a bug](#)

18.2.2.2. Non-clustered Instances

18.2.2.2.1. Launch a Non-clustered JBoss Enterprise Application Platform 6 Instance

Summary

This topic covers the steps required to launch a non-clustered instance of JBoss Enterprise Application Platform 6 on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- ▶ A suitable Red Hat AMI. Refer to [Section 18.1.6, “Supported Red Hat AMIs”](#).
- ▶ A pre-configured Security Group which allows incoming requests on at least ports 22, 8080, and 9990.

Procedure 18.1. Task

1. Configure the **User Data** field. The configurable parameters are available here: [Section 18.4.1, “Permanent Configuration Parameters”](#), [Section 18.4.2, “Custom Script Parameters”](#).

Example 18.1. Example User Data Field

The example shows the User Data field for a non-clustered JBoss Enterprise Application Platform 6 instance. The password for the user **admin** has been set to **adminpwd**.

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security
# reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"
# deploy /usr/share/java/jboss-ec2-eap-applications/<app name>.war
EOC

EOF
```

2. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```

 **Note**

yum -y update should be run regularly, to apply security fixes and enhancements.

3. Launch the Red Hat AMI instance.

Result

A non-clustered instance of JBoss Enterprise Application Platform 6 has been configured, and launched on a Red Hat AMI.

[Report a bug](#)

18.2.2.2.2. Deploy an Application on a non-clustered JBoss Enterprise Application Platform Instance

Summary

This topic covers deploying an application to a non-clustered JBoss Enterprise Application Platform 6 instance on a Red Hat AMI.

1. A. Deploy the Sample Application

Add the following lines to the **User Data** field:

```
# Deploy the sample application from the local filesystem  
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war
```

Example 18.2. Example User Data Field with Sample Application

This example uses the sample application provided on the Red Hat AMI. It also includes basic configuration for a non-clustered JBoss Enterprise Application Platform 6 instance. The password for the user **admin** has been set to **adminpwd**.

```
JBOSSAS_ADMIN_PASSWORD=adminpwd  
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces  
  
# In production, access to these ports needs to be restricted for  
# security reasons  
PORTS_ALLOWED="9990 9443"  
  
cat> $USER_SCRIPT << "EOF"  
  
# Create a file of CLI commands to be executed after starting the server  
cat> $USER_CLI_COMMANDS << "EOC"  
  
# Deploy the sample application from the local filesystem  
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war  
EOC  
  
EOF
```

B. Deploy a Custom Application

Add the following lines to the **User Data** field, configuring the application name and the URL:

```
# Get the application to be deployed from an Internet URL  
mkdir -p /usr/share/java/jboss-ec2-eap-applications  
wget https://<your secure storage hostname>/<path>/<app name>.war -O  
/usr/share/java/jboss-ec2-eap-applications/<app name>.war
```

Example 18.3. Example User Data Field with Custom Application

This example uses an application called **MyApp**, and includes basic configuration for a non-clustered JBoss Enterprise Application Platform 6 instance. The password for the user **admin** has been set to **adminpwd**.

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for
# security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://PATH_TO_MYAPP/MyApp.war -O /usr/share/java/jboss-ec2-eap-
applications/MyApp.war

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"
deploy /usr/share/java/jboss-ec2-eap-applications/MyApp.war
EOC

EOF
```

2. Launch the Red Hat AMI instance.

Result

The application has been successfully deployed to JBoss Enterprise Application Platform 6.

[Report a bug](#)

18.2.2.3. Test the Non-clustered JBoss Enterprise Application Platform 6 Instance**Summary**

This topic covers the steps required to test that the non-clustered JBoss Enterprise Application Platform 6 is running correctly.

Procedure 18.2. Task

1. Determine the instance's **Public DNS**, located in the instance's details pane.
2. Navigate to **http://<public-DNS>:8080**.
3. Confirm that the JBoss Enterprise Application Platform home page appears, including a link to the Admin console. If the home page is not available, refer here: [Section 18.5.1, "About Troubleshooting Amazon EC2"](#).
4. Click on the **Admin Console** hyperlink.
5. Log in:
 - » Username: **admin**
 - » Password: Specified in the **User Data** field here: [Section 18.2.2.1, "Launch a Non-clustered JBoss Enterprise Application Platform 6 Instance"](#).
6. **Test the Sample Application**
Navigate to **http://<public-DNS>:8080/hello** to test that the sample application is running successfully. The text **Hello World!** should appear in the browser. If the text is not visible, refer here: [Section 18.5.1, "About Troubleshooting Amazon EC2"](#).
7. Log out of the JBoss Enterprise Application Platform Admin Console.

Result

The JBoss Enterprise Application Platform 6 instance is running correctly.

[Report a bug](#)

18.2.2.3. Non-clustered Managed Domains

18.2.2.3.1. Launch an Instance to Serve as a Domain Controller

Summary

This topic covers the steps required to launch a non-clustered JBoss Enterprise Application Platform 6 managed domain on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- ▶ A suitable Red Hat AMI. Refer to [Section 18.1.6, “Supported Red Hat AMIs”](#).
- ▶ [Section 18.2.3.4, “Create a Virtual Private Cloud \(VPC\)”](#)
- ▶ [Section 18.2.3.5, “Launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the VPC”](#)
- ▶ [Section 18.2.3.6, “Configure the VPC Private Subnet Default Route”](#)

Procedure 18.3. Task

1. In the Security Group tab, ensure all traffic is allowed. Red Hat Enterprise Linux's built-in firewall capabilities can be used to restrict access if desired.
2. Set the public subnet of the VPC to *running*.
3. Select a static IP.
4. Configure the **User Data** field. The configurable parameters are available here: [Section 18.4.1, “Permanent Configuration Parameters”](#), [Section 18.4.2, “Custom Script Parameters”](#).

Example 18.4. Example User Data Field

The example shows the User Data field for a non-clustered JBoss Enterprise Application Platform 6 managed domain. The password for the user **admin** has been set to **admin**.

```
## password that will be used by slave host controllers to connect to the
domain controller
JBOSSAS_ADMIN_PASSWORD=admin

## subnet prefix this machine is connected to
SUBNET=10.0.0.

##### to run the example no modifications below should be needed #####
JBoss_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBoss_IP=`hostname | sed -e 's/ip\-\// -e 'y\-\./`\ #listen on
public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"

# Add the modcluster subsystem to the default profile to set up a proxy
/profile=default/subsystem=web/connector=ajp:add(name=ajp,protocol=AJP/1.3,
scheme=http,socket-binding=ajp)
/:composite(steps=[ {"operation" => "add", "address" => [ ("profile" =>
"default"), ("subsystem" => "modcluster") ] }, { "operation" => "add",
"address" => [ ("profile" => "default"), ("subsystem" => "modcluster"),
("mod-cluster-config" => "configuration") ], "advertise" => "false",
"proxy-list" => "${jboss.modcluster.proxyList}", "connector" => "ajp"}, { "operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster"), ("mod-cluster-config" => "configuration"),
("dynamic-load-provider" => "configuration") ] }, { "operation" => "add",
"address" => [ ("profile" => "default"), ("subsystem" => "modcluster"),
("mod-cluster-config" => "configuration"), ("dynamic-load-provider" =>
"configuration"), ("load-metric" => "busyness") ], "type" => "busyness"} ])

# Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/hello.war --server-
groups=main-server-group
EOC

## this will workaround the problem that in a VPC, instance hostnames are
not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-$SUBNET//.-$i" ;
done >> /etc/hosts

EOF
```

5. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```

 **Note**

yum -y update should be run regularly, to apply security fixes and enhancements.

6. Launch the Red Hat AMI instance.**Result**

A non-clustered JBoss Enterprise Application Platform 6 managed domain has been configured, and launched on a Red Hat AMI.

[Report a bug](#)

18.2.2.3.2. Launch One or More Instances to Serve as Host Controllers

Summary

This topic covers the steps required to launch one or more instances of JBoss Enterprise Application Platform 6 to serve as non-clustered host controllers on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- » Configure and launch the non-clustered domain controller. Refer to [Section 18.2.2.3.1, “Launch an Instance to Serve as a Domain Controller”](#).

Procedure 18.4. Launch Host Controllers

For each instance you would like to create, repeat the following steps:

1. Select an AMI.
2. Define the desired number of instances (the number of slave host controllers).
3. Select the VPC and instance type.
4. Click on Security Group.
5. Ensure that all traffic from the JBoss Enterprise Application Platform subnet is allowed.
6. Define other restrictions as desired.
7. Add the following into the User Data field:

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## host controller setup
JBoss_DOMAIN_MASTER_ADDRESS=10.0.0.5
JBoss_HOST_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.1.

##### to run the example no modifications below should be needed #####
JBoss_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBoss_IP=`hostname | sed -e 's/ip-//' -e 'y/-//` #listen on public/private
EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/other-server-group/main-server-group/" \
$JBoss_CONFIG_DIR/$JBoss_HOST_CONFIG

## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "${SUBNET}${i}\tip-${SUBNET}..-$i" ;
done >> /etc/hosts

EOF
```

8. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```

Note

yum -y update should be run regularly, to apply security fixes and enhancements.

9. Launch the Red Hat AMI instance.

Result

The JBoss Enterprise Application Platform 6 non-clustered host controllers are configured and launched on a Red Hat AMI.

[Report a bug](#)

18.2.2.3.3. Test the Non-Clustered JBoss Enterprise Application Platform 6 Managed Domain

Summary

This topic covers the steps required to test the non-clustered JBoss Enterprise Application Platform managed domain on a Red Hat AMI (Amazon Machine Image).

To test the Managed Domain you must know the elastic IP addresses of both the Apache HTTPD and JBoss Enterprise Application Platform Domain Controller.

Prerequisites

- ▶ Configure and launch the domain controller. Refer to [Section 18.2.2.3.1, “Launch an Instance to Serve as a Domain Controller”](#).
- ▶ Configure and launch the host controllers. Refer to [Section 18.2.2.3.2, “Launch One or More Instances to Serve as Host Controllers”](#).

Procedure 18.5. Test the Web Server

- ▶ Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD** in a browser to confirm the web server is running successfully.

Procedure 18.6. Test the Domain Controller

1. Navigate to **http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console**
2. Log in using the username of **admin** and the password specified in the User Data field for the domain controller and the admin console landing page for a managed domain should appear (**http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances**).
3. Click the Server label at the top right side of the screen, and select any of the host controllers in the Host dropdown menu at the top left side of the screen.
4. Verify that each host controller has two server configurations called **server-one** and **server-two** and that they both belong to the **main-server-group**.
5. Log out of the JBoss Enterprise Application Platform Admin Console.

Procedure 18.7. Test the Host Controllers

1. Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD/hello** to test that the sample application is running successfully. The text **Hello World!** should appear in the browser.
If the text is not visible, refer here: Section 18.5.1, “About Troubleshooting Amazon EC2”.
2. Connect to the Apache HTTPD instance:

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTPD
```

3. Navigate to **http://localhost:7654/mod_cluster-manager** to confirm all instances are running correctly.

Result

The JBoss Enterprise Application Platform 6 web server, domain controller, and host controllers are running correctly on a Red Hat AMI.

[Report a bug](#)

18.2.3. Clustered JBoss Enterprise Application Platform 6

18.2.3.1. About Clustered Instances

A clustered instance is an Amazon EC2 instance running JBoss Enterprise Application Platform 6 with clustering enabled. Another instance running Apache HTTPD will be acting as the proxy for the instances in the cluster.

The JBoss Enterprise Application Platform 6 AMIs include two configuration files for use in clustered instances, `standalone-ec2-ha.xml` and `standalone-mod_cluster-ec2-ha.xml`. Each of these configuration files provides clustering without the use of multicast because Amazon EC2 does not support multicast. This is done by using TCP unicast for cluster communications and S3_PING as the discovery protocol. The `standalone-mod_cluster-ec2-ha.xml` configuration also provides easy registration with mod_cluster proxies.

Similarly, the `domain-ec2.xml` configuration file provides two profiles for use in clustered managed domains: ec2-ha, and mod_cluster-ec2-ha.

[Report a bug](#)

18.2.3.2. Create a Relational Database Service Database Instance

Summary

This topic covers the steps to create a relational database service database instance, using MySQL as an example.



Warning

It is highly recommended that the back-up and maintenance features remain enabled for production environments.



Important

It is good practice to create separate user/password pairs for each application accessing the database. Tune other configuration options according to your application's requirements.

Procedure 18.8. Task

1. Click on the **RDS** in the AWS console.
2. Subscribe to the service if needed.
3. Click on **Launch DB instance**.
4. Click on **MySQL**.
 - a. Select a version. For example, **5.5.12**.
 - b. Select **small instance**.
 - c. Ensure **Multi-AZ Deployment** and **Auto upgrade** are **off**.
 - d. Set **Storage** to **5GB**.
 - e. Define the database administrator's username and password and click **Next**.
 - f. Select a database name to be created with the instance, and click **Next**.
 - g. Disable back-ups and maintenance, if necessary.
 - h. Confirm the settings.

Result

The database is created. It will initialize and be ready for use after a few minutes.

[Report a bug](#)

18.2.3.3. About Virtual Private Clouds

An Amazon Virtual Private Cloud (Amazon VPC) is a feature of Amazon Web Services (AWS) that allows

you to isolate a set of AWS resources in a private network. The topology and configuration of this private network can be customized to your needs.

Refer to the Amazon Virtual Private Cloud website for more information <http://aws.amazon.com/vpc/>.

[Report a bug](#)

18.2.3.4. Create a Virtual Private Cloud (VPC)

Summary

This topic covers the steps required to create a Virtual Private Cloud, using a database external to the VPC as an example. Your security policies may require connection to the database to be encrypted. Please refer to Amazon's *RDS FAQ* for details about encrypting the database connections.



Important

VPC is recommended for a JBoss Enterprise Application Platform cluster setup as it greatly simplifies secure communication between cluster nodes, a JON Server and the mod_cluster proxy. Without a VPC, these communication channels need to be encrypted and authenticated. For detailed instructions on configuring SSL, refer here: [Section 13.2.3, "Implement SSL Encryption for the JBoss Enterprise Application Platform Web Server"](#).

1. Go to the VPC tab in the AWS console.
2. Subscribe to the service if needed.
3. Click on "**Create new VPC**".
4. Choose a VPC with one public and one private subnet.
 - a. Set the public subnet to be **10.0.0.0/24**.
 - b. Set the private subnet to be **10.0.1.0/24**.
5. Go to **Elastic IPs**.
6. Create an elastic IP for use by the mod_cluster proxy/NAT instance.
7. Go to **Security groups** and create a security group to allow all traffic in and out.
8. Go to Network ACLs
 - a. Create an ACL to allow all traffic in and out.
 - b. Create an ACL to allow all traffic out and traffic in on only TCP ports **22, 8009, 8080, 8443, 9443, 9990** and **16163**.

Result

The Virtual Private Cloud has been successfully created.

[Report a bug](#)

18.2.3.5. Launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the VPC

Summary

This topic covers the steps required to launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the Virtual Private Cloud.

Prerequisites

- ▶ [Section 18.2.3.2, "Create a Relational Database Service Database Instance"](#).
- ▶ [Section 18.2.3.4, "Create a Virtual Private Cloud \(VPC\)"](#)

Procedure 18.9. Launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the VPC

1. Create an elastic IP for this instance.
2. Select an AMI.
3. Go to **Security Group** and allow all traffic (use Red Hat Enterprise Linux's built-in firewall capabilities to restrict access if desired).
4. Select "**running**" in the public subnet of the VPC.

5. Select a static IP (e.g. **10.0.0.4**).
6. Put the following in the **User Data** field:

```
JBOSSCONF=disabled

cat > $USER_SCRIPT << "EOS"

echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/eth0/rp_filter

iptables -I INPUT 4 -s 10.0.1.0/24 -p tcp --dport 7654 -j ACCEPT
iptables -I INPUT 4 -p tcp --dport 80 -j ACCEPT

iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -I FORWARD -s 10.0.1.0/24 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 ! -s 10.0.0.4 -j MASQUERADE

# balancer module incompatible with mod_cluster
sed -i -e 's/LoadModule proxy_balancer_module/#\0/' /etc/httpd/conf/httpd.conf

cat > /etc/httpd/conf.d/mod_cluster.conf << "EOF"
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so

Listen 7654

# workaround JBPAPP-4557
MemManagerFile /var/cache/mod_proxy/manager

<VirtualHost *:7654>
    <Location /mod_cluster-manager>
        SetHandler mod_cluster-manager
        Order deny,allow
        Deny from all
        Allow from 127.0.0.1
    </Location>

    <Location />
        Order deny,allow
        Deny from all
        Allow from 10.
        Allow from 127.0.0.1
    </Location>

    KeepAliveTimeout 60
    MaxKeepAliveRequests 0
    ManagerBalancerName mycluster
    ServerAdvertise Off
    EnableMCPMReceive On
</VirtualHost>
EOF

echo "`hostname | sed -e 's/ip-//' -e 'y/-/./'`" >> /etc/hosts

semanage port -a -t http_port_t -p tcp 7654 #add port in the apache port list
for the below to work
setsebool -P httpd_can_network_relay 1 #for mod_proxy_cluster to work
chcon -t httpd_config_t -u system_u /etc/httpd/conf.d/mod_cluster.conf

##### Uncomment the following line when launching a managed domain #####
# setsebool -P httpd_can_network_connect 1

service httpd start

EOS
```

7. Disable the Amazon EC2 cloud source/destination checking for this instance so it can act as a router.

- a. Right-click on the running Apache HTTPD instance and choose "**Change Source/Dest check**".
 - b. Click on **Yes, Disable**.
8. Assign the elastic IP to this instance.

Result

The Apache HTTPD instance has been launched successfully.

[Report a bug](#)

18.2.3.6. Configure the VPC Private Subnet Default Route

Summary

This topic covers the steps required to configure the VPC private subnet default route. The JBoss Enterprise Application Platform cluster nodes will run in the private subnet of the VPC, but cluster nodes require Internet access for S3 connectivity. A default route needs to be set to go through the NAT instance.

Procedure 18.10. Configure the VPC Private Subnet Default Route

1. Navigate to the Apache HTTPD instance in the Amazon AWS console.
2. Navigate to the **VPC → route tables**.
3. Click on the routing table used by the private subnet.
4. In the field for a new route enter **0.0.0.0/0**.
5. Click on "**Select a target**".
6. Select "**Enter Instance ID**".
7. Choose the ID of the running Apache HTTPD instance.

Result

The default route has been successfully configured for the VPC subnet.

[Report a bug](#)

18.2.3.7. About Identity and Access Management (IAM)

Identity and Access Management (IAM) provides configurable security for your AWS resources. IAM can be configured to use accounts created in IAM or to provide identity federation between IAM and your own identity services.

Refer to the AWS Identity and Access Management website for more information
<http://aws.amazon.com/iam/>.

[Report a bug](#)

18.2.3.8. Configure IAM Setup

Summary

This topic covers the configuration steps required for setting up IAM for clustered JBoss Enterprise Application Platform instances. The **S3_PING** protocol uses an S3 bucket to discover other cluster members. **JGroups** version 3.0.x requires Amazon AWS account access and secret keys to authenticate against the S3 service.

It is a security risk to enter your main account credentials in the user-data field, store them online or in an AMI. To circumvent this, a separate account can be created using the Amazon IAM feature which would be only granted access to a single S3 bucket.

Procedure 18.11. Configure IAM Setup

1. Go to the IAM tab in the AWS console.
2. Click on **users**.
3. Select **Create New Users**.
4. Choose a name, and ensure the **Generate an access key for each User** option is checked.

5. Select **Download credentials**, and save them in a secure location.
6. Close the window.
7. Click on the newly created user.
8. Make note of the **User ARM** value. This value is required to set up the S3 bucket, documented here: [Section 18.2.3.10, "Configure S3 Bucket Setup"](#).

Result

The IAM user account has been successfully created.

[Report a bug](#)

18.2.3.9. About the S3 Bucket

S3 Buckets are the basic organization store unit in the Amazon Simple Storage System (Amazon S3). A bucket can store any number of arbitrary objects and must have a unique name to identify it with Amazon S3..

Refer to the Amazon S3 website for more information, <http://aws.amazon.com/s3/>.

[Report a bug](#)

18.2.3.10. Configure S3 Bucket Setup

Summary

This topic covers the steps required to configure a new S3 bucket.

Prerequisites

- » [Section 18.2.3.8, "Configure IAM Setup"](#).

Procedure 18.12. Configure S3 Bucket Setup

1. Open the **S3** tab in the AWS console.
2. Click on **Create Bucket**.
3. Choose a name for the bucket and click **Create**.
4. Right click on the new bucket and select **Properties**.
5. Click **Add bucket policy** in the permissions tab.
6. Click **New policy** to open the policy creation wizard.
 - a. Copy the following content into the new policy, replacing **arn:aws:iam::055555555555:user/jbosscluster*** with the value defined here: [Section 18.2.3.8, "Configure IAM Setup"](#). Change both instances of **clusterbucket123** to the name of the bucket defined in step 3 of this procedure.

Note

Bucket names are unique across the entire S3. Names cannot be reused.

```
{
    "Version": "2008-10-17",
    "Id": "Policy1312228794320",
    "Statement": [
        {
            "Sid": "Stmt1312228781799",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::055555555555:user/jbosscluster"
                ]
            },
            "Action": [
                "s3>ListBucketVersions",
                "s3>GetObjectVersion",
                "s3>ListBucket",
                "s3>PutBucketVersioning",
                "s3>DeleteObject",
                "s3>DeleteObjectVersion",
                "s3>GetObject",
                "s3>ListBucketMultipartUploads",
                "s3>ListMultipartUploadParts",
                "s3>PutObject",
                "s3>GetBucketVersioning"
            ],
            "Resource": [
                "arn:aws:s3:::clusterbucket123/*",
                "arn:aws:s3:::clusterbucket123"
            ]
        }
    ]
}
```

Result

A new S3 bucket has been created, and configured successfully.

[Report a bug](#)

18.2.3.11. Clustered Instances

18.2.3.11.1. Launch Clustered JBoss Enterprise Application Platform 6 AMIs

Summary

This topic covers the steps required to launch clustered JBoss Enterprise Application Platform 6 AMIs.

Prerequisites

- ▶ [Section 18.2.3.2, “Create a Relational Database Service Database Instance”](#).
- ▶ [Section 18.2.3.4, “Create a Virtual Private Cloud \(VPC\)”](#).
- ▶ [Section 18.2.3.5, “Launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the VPC”](#).
- ▶ [Section 18.2.3.6, “Configure the VPC Private Subnet Default Route”](#).
- ▶ [Section 18.2.3.8, “Configure IAM Setup”](#).
- ▶ [Section 18.2.3.10, “Configure S3 Bucket Setup”](#).



Warning

Running JBoss Enterprise Application Platform cluster in a subnet with network mask smaller than 24 bits or spanning multiple subnets complicates acquiring a unique server peer ID for each cluster member.

Refer to the **JBOSS_CLUSTER_ID** variable for information on how to make such a configuration work reliably: [Section 18.4.1, “Permanent Configuration Parameters”](#).



Important

The auto-scaling Amazon EC2 feature can be used with JBoss Enterprise Application Platform cluster nodes. However, ensure it is tested **before** deployment. You should ensure that your particular workloads scale to the desired number of nodes, and that the performance meets your needs for the instance type you are planning to use (different instance types receive a different share of the EC2 cloud resources).

Furthermore, instance locality and current network/storage/host machine/RDS utilization can affect performance of a cluster. Test with your expected real-life loads and try to account for unexpected conditions.



Down-scaling a cluster

The Amazon EC2 *scale-down* action terminates the nodes without any chance to gracefully shut down, and, as some transactions might be interrupted, other cluster nodes (and load balancers) will need time to fail over. This is likely to impact your application users' experience.

It is recommended that you either scale down the application cluster manually by disabling the server from the mod_cluster management interface until processed sessions are completed, or shut down the JBoss Enterprise Application Platform instance gracefully (SSH access to the instance or JON can be used).

Test that your chosen procedure for scaling-down does not lead to adverse effects on your users' experience. Additional measures might be required for particular workloads, load balancers and setups.

Procedure 18.13. Launch Clustered JBoss Enterprise Application Platform 6 AMIs

1. Select an AMI.
2. Define the desired number of instances (the cluster size).
3. Select the VPC and instance type.
4. Click on **Security Group**.
5. Ensure that all traffic from the JBoss Enterprise Application Platform cluster subnet is allowed.
6. Define other restrictions as desired.
7. Add the following into the **User Data** field:

Example 18.5. Example User Data Field

```

## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## password to access admin console
JBOSSAS_ADMIN_PASSWORD=<your password for opening admin console>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

##### to run the example no modifications below should be needed #####
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-./` #listen on
public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-* .jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"

<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy sample application from local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war

## ExampleDS configuration for MySQL database
data-source remove --name=ExampleDS
/subsystem=datasources/jdbc-driver=mysql:add(driver-name="mysql",driver-
module-name="com.mysql")
data-source add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/subsystem=datasources/data-source=ExampleDS:enable
/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
EOC

## this will workaround the problem that in a VPC, instance hostnames are
not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "${SUBNET}${i}\tip-${SUBNET//.-}${i}" ;
done >> /etc/hosts

EOF

```

Result

The clustered JBoss Enterprise Application Platform 6 AMIs have been configured and launched successfully.

[Report a bug](#)

18.2.3.11.2. Test the Clustered JBoss Enterprise Application Platform 6 Instance

Summary

This topic covers the steps to confirm that the clustered JBoss Enterprise Application Platform 6 instances are running correctly.

Procedure 18.14. Testing the Clustered Instance

1. Navigate to http://ELASTIC_IP_OF_APACHE_HTTPD in a browser to confirm the web server is running successfully.

2. Test the Clustered Nodes

- a. Navigate to http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/put.jsp in a browser.
- b. Verify that one of the cluster nodes logs the following message:

Putting date now

- c. Stop the cluster node that logged the message in the previous step.
- d. Navigate to http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/get.jsp in a browser.
- e. Verify that the time shown is the same as the time that was PUT using `put.jsp` in Step 2-a.
- f. Verify that one of the running cluster nodes logs the following message:

Getting date now

- g. Restart the stopped clustered node.
- h. Connect to the Apache HTTPD instance:

`ssh -L7654:localhost:7654 <ELASTIC_IP_OF_APACHE_HTTPD>`

- i. Navigate to http://localhost:7654/mod_cluster-manager to confirm all instances are running correctly.

Result

The clustered JBoss Enterprise Application Platform 6 instance have been tested, and confirmed to be working correctly.

[Report a bug](#)

18.2.3.12. Clustered Managed Domains

18.2.3.12.1. Launch an Instance to Serve as a Cluster Domain Controller

Summary

This topic covers the steps required to launch a clustered JBoss Enterprise Application Platform 6 managed domain on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- » A suitable Red Hat AMI. Refer to [Section 18.1.6, “Supported Red Hat AMIs”](#).
- » [Section 18.2.3.4, “Create a Virtual Private Cloud \(VPC\)”](#)
- » [Section 18.2.3.5, “Launch an Apache HTTPD instance to serve as a mod_cluster proxy and a NAT instance for the VPC”](#)
- » [Section 18.2.3.6, “Configure the VPC Private Subnet Default Route”](#)

Procedure 18.15. Launch a Cluster Domain Controller

1. Create an elastic IP for this instance.
2. Select an AMI.
3. Go to Security Group and allow all traffic (use Red Hat Enterprise Linux's built-in firewall capabilities to restrict access if desired).
4. Choose "running" in the public subnet of the VPC.
5. Choose a static IP (e.g. 10.0.0.5).
6. Put the following in the User Data: field:

```

## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## password that will be used by slave host controllers to connect to the
domain controller
JBOSAS_ADMIN_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.0.

##### to run the example no modifications below should be needed #####
JBoss_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBoss_IP=`hostname | sed -e 's/ip-//` -e 'y/-./` #listen on public/private
EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbosas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-* .jar
/usr/share/jbosas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbosas/modules/com/mysql/main/module.xml <<"EOM"

<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war --server-
groups=other-server-group

## ExampleDS configuration for MySQL database
data-source --profile=mod_cluster-ec2-ha remove --name=ExampleDS
/profile=mod_cluster-ec2-ha/subsystem=datasources/jdbc-
driver=mysql:add(driver-name="mysql",driver-module-name="com.mysql")
data-source --profile=mod_cluster-ec2-ha add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/profile=mod_cluster-ec2-ha/subsystem=datasources/data-
source=ExampleDS:enable
EOC

## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "${SUBNET}${i}\tip-${SUBNET//./-}${i}" ;
done >> /etc/hosts

EOF

```

7. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```

 **Note**

yum -y update should be run regularly, to apply security fixes and enhancements.

8. Launch the Red Hat AMI instance.

Result

A clustered JBoss Enterprise Application Platform 6 managed domain is configured and launched on a Red Hat AMI.

[Report a bug](#)

18.2.3.12.2. Launch One or More Instances to Serve as Cluster Host Controllers

Summary

This topic covers the steps required to launch one or more instances of JBoss Enterprise Application Platform 6 to serve as cluster host controllers on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- » Configure and launch the cluster domain controller. Refer to [Section 18.2.3.12.1, “Launch an Instance to Serve as a Cluster Domain Controller”](#).

Procedure 18.16. Launch Host Controllers

For each instance you would like to create, repeat the following steps:

1. Select an AMI.
2. Define the desired number of instances (the number of slave host controllers).
3. Select the VPC and instance type.
4. Click on Security Group.
5. Ensure that all traffic from the JBoss Enterprise Application Platform cluster subnet is allowed.
6. Define other restrictions as desired.
7. Add the following into the User Data field:

```

## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBoss_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBoss_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBoss_JGROUPS_S3_PING_BUCKET=<your bucket name>

## host controller setup
JBoss_DOMAIN_MASTER_ADDRESS=10.0.0.5
JBoss_HOST_PASSWORD=<password for slave host controllers>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

##### to run the example no modifications below should be needed #####
JBoss_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBoss_IP=`hostname | sed -e 's/ip-//' -e 'y/-//` #listen on public/private
EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/main-server-group/other-server-group/" $JBoss_CONFIG_DIR/$JBoss_HOST_CONFIG

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-* .jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"

<module xmlns="urn:jboss:module:1.0" name="com.mysql">
<resources>
    <resource-root path="mysql-connector-java.jar"/>
</resources>
<dependencies>
    <module name="javax.api"/>
</dependencies>
</module>
EOM

## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-$SUBNET//.-$i" ;
done >> /etc/hosts

EOF

```

8. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```

Note

yum -y update should be run regularly, to apply security fixes and enhancements.

9. Launch the Red Hat AMI instance.

Result

The JBoss Enterprise Application Platform 6 cluster host controllers are configured and launched on a Red Hat AMI.

[Report a bug](#)

18.2.3.12.3. Test the Clustered JBoss Enterprise Application Platform 6 Managed Domain

Summary

This topic covers the steps required to test the clustered JBoss Enterprise Application Platform managed domain on a Red Hat AMI (Amazon Machine Image).

To test the Managed Domain you must know the elastic IP addresses of both the Apache HTTPD and JBoss Enterprise Application Platform Domain Controller.

Prerequisites

- ▶ Configure and launch the cluster domain controller. Refer to [Section 18.2.3.12.1, “Launch an Instance to Serve as a Cluster Domain Controller”](#).
- ▶ Configure and launch the cluster host controllers. Refer to [Section 18.2.3.12.2, “Launch One or More Instances to Serve as Cluster Host Controllers”](#).

Procedure 18.17. Test the Apache HTTPD instance

- ▶ Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD** in a browser to confirm the web server is running successfully.

Procedure 18.18. Test the Domain Controller

1. Navigate to **http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console**
2. Log in using the username **admin** and the password specified in the User Data field for the domain controller. Once logged in, the admin console landing page for a managed domain should appear (http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances).
3. Click the Server label at the top right side of the screen. Select any of the host controllers in the Host dropdown menu at the top left side of the screen.
4. Verify that this host controller has two server configurations called **server-one** and **server-two** and verify that they both belong to the **other-server-group**.

Procedure 18.19. Test the Host Controllers

1. Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/put.jsp** in a browser.
2. Verify that one of the host controllers logs the following message: **Putting date now**.
3. Stop the server instance that logged the message in the previous step (see Section 2.8.3, Stop a Server Using the Management Console).
4. Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/get.jsp** in a browser.
5. Verify that the time shown is the same as the time that was **PUT** using **put.jsp** in Step 2.
6. Verify that one of the running server instances logs the following message: **Getting date now**.
7. Restart the stopped server instance (see Section 2.8.3, Start a Server Using the Management Console)
8. Connect to the Apache HTTPD instance.

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTPD
```

9. Navigate to **http://localhost:7654/mod_cluster-manager** to confirm all instances are running correctly.

Result

The JBoss Enterprise Application Platform 6 web server, domain controller, and host controllers are running correctly on a Red Hat AMI.

[Report a bug](#)

18.3. Establishing Monitoring with JBoss Operations Network (JON)

18.3.1. About AMI Monitoring

With your business application deployed to a correctly-configured AMI instance, the next step is to establish monitoring of the platform with JBoss Operations Network (JON).

The JON server is commonly located inside a corporate network, so it's necessary to establish a secure connection between the server and each of its agents. Establishing a VPN between the two points is the most common solution but this complicates the required networking configuration. This chapter provides network configuration guidelines for enabling communication between the JON agent and JON server. For more extensive information on configuration, management, and usage please refer to the official Red Hat documentation for JBoss Operations Network (JON).

Figure 18.1. JON Server connectivity

[Report a bug](#)

18.3.2. About Connectivity Requirements

Registering a JON agent with its servers requires two-way communication between agent and servers. The JON Agent needs access to port 7080 on all JON servers, except in the case of SSL where port 7443 is used. Each JON server must be able to access each of the connected agents on a unique host and port pairing. The agent port is usually 16163.

If there are multiple clustered JON servers, make sure each agent can communicate with all servers in the JON cluster via the IP and hostname pairs as configured through the JON server administration console. The JON server used by the agent to register may not be the server it tries to use after initialization.

[Report a bug](#)

18.3.3. About Network Address Translation (NAT)

A corporate VPN gateway acting in routed mode greatly simplifies network configuration. If your corporate VPN gateway is acting in NAT mode however, the JON server does not have direct visibility of agents. In this case, port forwarding needs to be configured for each agent.

NAT VPN configurations require a port on the gateway to be forwarded to the JON agent's address or port on the managed machine. The JON agent also needs to be configured to tell the server the forwarded port number and IP address. You can find further information in the `rhq.communications.connector.*` description for the `agent-configuration.xml`

configuration file.

[Report a bug](#)

18.3.4. About Amazon EC2 and DNS

JON servers and JON agents need to be able to resolve each others' hostnames. The DNS resolution is more complicated in the case of a VPN configuration. Connected servers have multiple possible options. One option is to use either the Amazon EC2 or the corporate network's DNS servers. Another option is to use a split DNS configuration where the corporate DNS servers are used for resolving names in particular domains, and the Amazon EC2 DNS servers are used for resolving all other names.

[Report a bug](#)

18.3.5. About Routing in EC2

All Amazon EC2 servers have a **source/destination checking** routing feature activated by default. This feature drops any packets being sent to the server which have a destination different from the machine's IP address. If the VPN solution selected for connecting agents to the JON server includes a router, this feature needs to be turned off for the server or servers acting as routers or VPN gateways. This configuration setting can be accessed via the Amazon AWS console. Disabled **source/destination checking** is also required in a Virtual Private Cloud (VPC).

Some VPN configurations route general Internet traffic through the corporate VPN by default. It is recommended that you avoid this as it may be a slower and less efficient configuration for your particular needs.

While the use of a proper addressing schema is not a concern specific to JON, poor schemas can affect it. Amazon EC2 assigns IP addresses from the 10.0.0.0/8 network. Instances usually have a public IP address also, but only network traffic on the internal IP address within the same availability zone is free. To avoid using the 10.0.0.0/8 network in private addressing, there are a few things to consider.

- ▶ When creating a VPC, avoid allocating addresses already in use in the private network to avoid connectivity problems.
- ▶ If an instance needs access to availability zone local resources, make sure Amazon EC2 private addresses are used and traffic is not routed through the VPN.
- ▶ If an Amazon EC2 instance will access a small subset of corporate private network addresses (for example only JON servers), only these addresses should be routed through the VPN. This increases security and lowers the chance of Amazon EC2 or private network address space collisions.

[Report a bug](#)

18.3.6. About Terminating and Restarting with JON

One of the benefits of a cloud environment is the ease by which you can terminate and launch a machine instance. You can also launch an instance identical to the initial one. This may cause issues if the new instance tries to register with JON servers using the same agent name as the previously running agent. If this happens the JON server will not allow an agent to reconnect with a missing or non-matching identification token.

To avoid this, ensure that terminated agents are removed from the JON inventory before trying to connect an agent with the same name or specify the correct identification token when starting new agent.

Another issue that you might encounter is when an agent machine is assigned a new VPN IP address that no longer matches the address recorded in the JON configuration. An example might include a machine that is restarted or where a VPN connection is terminated. In this case, it is recommended that you bind the JON agent's life cycle to the VPN connection's life cycle. If the connection drops, you can stop the agent. When the connection is restored again, update **JON_AGENT_ADDR** in `/etc/sysconfig/jon-agent-ec2` to reflect the new IP address and restart the agent.

Information on how to change the agent's IP address can be found in the Configuring JON Servers and Agents Guide available at https://access.redhat.com/knowledge/docs/JBoss_Operations_Network/.

If there are a high number of instances launched and/or terminated it can become impractical to add and remove them manually from the JON inventory. JON's scripting capabilities can be used for automate these steps. Refer to the JON documentation for further information.

[Report a bug](#)

18.3.7. Configure an Instance to Register with JBoss Operations Network

Use the following procedure to register a JBoss Enterprise Application Platform instance with JBoss Operations Network.

- For JBoss Enterprise Application Platform, add this to the User Data field.

```
JON_SERVER_ADDR=jon2.it.example.com
## if instance not already configured to resolve its hostname
JON_AGENT_ADDR=`ip addr show dev eth0 primary to 0/0 | sed -n 's#.*inet \([0-
9.]*)\(.*\)/*#\\1#p'`
PORTS_ALLOWED=16163
# insert other JON options when necessary, see Appendix I
```

[Report a bug](#)

18.4. User Script Configuration

18.4.1. Permanent Configuration Parameters

Summary

The following parameters can be used to influence the configuration and operation of JBoss Enterprise Application Platform. Their contents are written to **/etc/sysconfig/jbossas** and **/etc/sysconfig/jon-agent-ec2**.

Table 18.2. Configurable Parameters

Name	Description	Default
JBOSS_JGROUPS_S3_PING_A CESS_KEY	Amazon AWS user account access key for S3_PING discovery if clustering is used.	N/A
JBOSS_JGROUPS_S3_PING_S CRET_ACCESS_KEY	Amazon AWS user account secret access key.	N/A
JBOSS_JGROUPS_S3_PING_B UCKET	Amazon S3 bucket to be used for S3_PING discovery.	N/A
JBOSS_CLUSTER_ID	ID of cluster member nodes. Only used for clustering. Accepted values are (in order): <ul style="list-style-type: none"> ▶ A valid cluster ID number in the range 0 - 1023. ▶ A network interface name, where the last octet of the IP is used as the value. ▶ "S3" as a value would coordinate ID usage through the S3 bucket used by jgroups' S3_PING. It is recommended to use the last octet of the IP (the default) when all cluster nodes are located in the same 24 or more bit subnet (for example, in a VPC subnet). 	Last octet of eth0's IP address
MOD_CLUSTER_PROXY_LIST	Comma-delimited list of IPs/hostnames of mod_cluster proxies if mod_cluster is to be used.	N/A
PORTS_ALLOWED	List of incoming ports to be allowed by firewall in addition to the default ones.	N/A
JBOSSAS_ADMIN_PASSWORD	Password for the admin user.	N/A
JON_SERVER_ADDR	JON server hostname or IP with which to register. This is only used for registration, after that the agent may communicate with other servers in the JON cluster.	N/A
JON_SERVER_PORT	Port used by the agent to communicate with the server.	7080
JON_AGENT_NAME	Name of JON agent, must be unique.	Instance's ID
JON_AGENT_PORT	Port that the agent listens on.	16163
JON_AGENT_ADDR	IP address that the JON agent is to be bound to. This is used when the server has more than one public address, (e.g. VPN).	JON agent chooses the IP of local hostname by default.
JON_AGENT_OPTS	Additional JON agent system properties which can be used for configuring SSL, NAT and other advanced settings.	N/A
JBOSS_SERVER_CONFIG	Name of JBoss EAP server configuration file to use. If JBOSS_DOMAIN_CONTROLLED=true, then domain-ec2.xml is used. Otherwise: <ul style="list-style-type: none"> ▶ If S3 config is present, then 	standalone.xml , standalone-full.xml , standalone-ec2-ha.xml , standalone-mod_cluster-ec2-ha.xml , domain-ec2.xml depending on the other parameters

	<p>standalone-ec2-ha.xml is used.</p> <ul style="list-style-type: none"> ▶ If MOD_CLUSTER_PROXY_LIST is specified, then standalone-mod_cluster-ec2-ha.xml is selected. ▶ If neither of the first two options are used, then the standalone.xml file is used. ▶ Can also be set to standalone-full.xml. 	other parameters.
JAVA_OPTS	Custom values to be added to the variable before JBoss Enterprise Application Platform starts.	JAVA_OPTS is built from the values of other parameters.
JBOSS_IP	IP address that the server is to be bound to.	127.0.0.1
JBOSSCONF	The name of the JBoss Enterprise Application Platform 6 profile to start. To prevent JBoss Enterprise Application Platform 6 from starting, JBOSSCONF can be set to disabled	standalone
JBOSS_DOMAIN_CONTROLLER	Sets whether or not this instance will run as a domain controller.	false
JBOSS_DOMAIN_MASTER_ADDRESS	IP address of remote domain controller.	N/A
JBOSS_HOST_NAME	The logical host name (within the domain). This needs to be distinct.	The value of the HOSTNAME environment variable.
JBOSS_HOST_USERNAME	The username the host controller should use when registering with the domain controller. If not provided, the JBOSS_HOST_NAME is used instead.	JBOSS_HOST_NAME
JBOSS_HOST_PASSWORD	The password the host controller should use when registering with the domain controller.	N/A
JBOSS_HOST_CONFIG	If JBOSS_DOMAIN_CONTROLLER=true, then host-master.xml or host-slave.xml , depending on the other parameters. If JBOSS_DOMAIN_MASTER_ADDRESS is present, then host-slave.xml is used.	host-master.xml or host-slave.xml , depending on the other parameters.

[Report a bug](#)

18.4.2. Custom Script Parameters

Summary

The following parameters can be used in the user customization section of the **User Data:** field.

Table 18.3. Configurable Parameters

Name	Description
JBOSS_DEPLOY_DIR	Deploy directory of the active profile (for example, <code>/var/lib/jbossas/standalone/deployments/</code>). Deployable archives placed in this directory will be deployed. Red Hat recommends using the Management Console or CLI tool to manage deployments instead of using the deploy directory.
JBOSS_CONFIG_DIR	Config directory of the active profile (for example, <code>/var/lib/jbossas/standalone/configuration</code>).
JBOSS_HOST_CONFIG	Name of the active host configuration file (for example, <code>host-master.xml</code>). Red Hat recommends using the Management Console or CLI tools to configure the server instead of editing the configuration file.
JBOSS_SERVER_CONFIG	Name of the active server configuration file (for example, <code>standalone-ec2-ha.xml</code>). Red Hat recommends using the Management Console or CLI tools to configure the server instead of editing the configuration file.
USER_SCRIPT	Path to the custom configuration script, which is available prior to sourcing user-data configuration.
USER_CLI_COMMANDS	Path to a custom file of CLI commands, which is available prior to sourcing user-data configuration.

[Report a bug](#)

18.5. Troubleshooting

18.5.1. About Troubleshooting Amazon EC2

EC2 does not provide any method out of the box to indicate an instance has started correctly and services are running properly. Use of an external system for monitoring and management is recommended. JBoss Operations Network (JON) can automatically discover, monitor and manage many services on an EC2 instance with the JON agent installed, including JBoss Enterprise Application Platform and its services, Tomcat, Httpd, PostgreSQL, etc. Since there's no difference between an EC-hosted or locally-hosted instance of JBoss Enterprise Application Platform, established JON monitoring of both types of deployments is identical.

[Report a bug](#)

18.5.2. Diagnostic Information

In case of a problem being detected by the JBoss Operations Network, Amazon CloudWatch or manual inspection, common sources of diagnostic information are:

- ▶ `/var/log/jboss-user-data.out` is the output of the jboss-ec2-eap init script and user custom configuration script.
- ▶ `/var/cache/jboss-ec2-eap/` contains the actual user data, custom script, and custom CLI commands used at instance start-up.
- ▶ `/var/log` also contains all the logs collected from machine start up, JBoss Enterprise Application Platform, httpd and most other services.

Access to these files is only available via an SSH session. Refer to the Amazon EC Getting Started Guide for details on how to configure and establish an SSH session with an Amazon EC2 instance.

[Report a bug](#)

Chapter 19. Supplemental References

19.1. Download Files From the Red Hat Customer Portal

Task Prerequisites

Before you begin this task, you need a Customer Portal account. Browse to <https://access.redhat.com> and click the **Register** link in the upper right corner to create an account.

Procedure 19.1. Task:

1. Browse to <https://access.redhat.com> and click the **Log in** link in the top right corner. Enter your credentials and click **Log In**.

Result:

You are logged into RHN and you are returned to the main web page at <https://access.redhat.com>.

2. **Navigate to the Downloads page.**

Use one of the following options to navigate to the **Downloads** page.

- A. Click the **Downloads** link in the top navigation bar.
- B. Navigate directly to <https://access.redhat.com/downloads/>.

3. **Select the product and version to download.**

Use one of the following ways to choose the correct product and version to download.

- A. Step through the navigation one level at a time.
- B. Search for your product using the search area at the top right-hand side of the screen.

4. **Download the appropriate file for your operating system and installation method of choice.**

Depending on the product you choose, you may have the choice of a Zip archive, RPM, or native installer for a specific operating system and architecture. Click either the file name or the **Download** link to the right of the file you want to download.

Result:

The file is downloaded to your computer.

[Report a bug](#)

19.2. Configure the Default JDK on Red Hat Enterprise Linux

Task Summary

It is possible to have multiple Java Development Kits (JDKs) installed on your Red Hat Enterprise Linux system. This task shows you how to specify which one your current environment uses. It uses the **alternatives** command.

This task only applies to Red Hat Enterprise Linux.



Note

It may not be necessary to do this step. Red Hat Enterprise Linux uses OpenJDK 1.6 as the default option. If this is what you want, and your system is working properly, you do not need to manually specify which JDK to use.

Task Prerequisites

In order to complete this task, you need to have superuser access, either through direct login or by means of the **sudo** command.

1. **Determine the paths for your preferred java and javac binaries.**

You can use the command **rpm -q1 packagename |grep bin** to find the locations of binaries installed from RPMs. The default locations of the **java** and **javac** binaries on Red Hat Enterprise Linux 32-bit systems are as follows.

Table 19.1. Default locations for java and javac binaries

JDK	Path
OpenJDK 1.6	/usr/lib/jvm/jre-1.6.0- openjdk/bin/java /usr/lib/jvm/java-1.6.0- openjdk/bin/javac
Oracle JDK 1.6	/usr/lib/jvm/jre-1.6.0- sun/bin/java /usr/lib/jvm/java-1.6.0- sun/bin/javac

2. Set the alternative you wish to use for each.

Run the following commands to set your system to use a specific **java** and **javac**:

/usr/sbin/alternatives --config java or **/usr/sbin/alternatives --config javac**. Follow the on-screen instructions.

3. Optional: Set the **java_sdk_1.6.0 alternative choice.**

If you want to use Oracle JDK, you need to set the alternative for **java_sdk_1.6.0**. as well. Use the following command: **/usr/sbin/alternatives --config java_sdk_1.6.0**. The correct path is usually **/usr/lib/jvm/java-1.6.0-sun**. You can do a file listing to verify it.

Result:

The alternative JDK is selected and active.

[Report a bug](#)

Revision History

Revision 0.3-9.400	2013-10-30	Rüdiger Landmann
Rebuild with publican 4.0.0		
Revision 0.3-9	Fri Apr 05 2013	Tom Wells
Backported the fix for a regression in Configure the JGroups Subsystem to Use TCP.		
Revision 0.3-6	Fri Mar 01 2013	Tom Wells
Corrected bugzilla issues and added order of deployment configuration topic.		
Revision 0.3-5	Fri Mar 01 2013	Tom Wells
Added Datasource and Resource Adapter Statistics.		
Revision 0.3-4	Thu Feb 28 2013	Tom Wells
Updated build to include bugzilla fixes and new introductory content.		
Revision 0.3-3	Tue Feb 26 2013	Tom Wells
Updated to include the managed domain Amazon content.		
Revision 0.3-2	Tue Feb 26 2013	Tom Wells
Updated to include several new bugzilla fixes.		
Revision 0.3-1	Wed Feb 20 2013	Tom Wells
Updated the build to include the Message Replication section.		
Revision 0.2-25	Mon Feb 11 2013	Tom Wells
Updating the JBoss Enterprise Application Platform 6 Administration and Configuration Guide. Several screenshots were corrected, and a number of Bugzilla issues were closed.		
Revision 0.2-24	Wed Feb 06 2013	Tom Wells
Fixed a regression in the TCP JGroups topics.		
Revision 0.2-23	Fri Jan 18 2013	Tom Wells
Corrected topic revision numbers for Amazon EC2 topics, updating them to the latest versions.		
Revision 0.2-22	Tue Jan 15 2013	Tom Wells
Updated Administration and Configuration Guide. Includes the Amazon EC2 Managed Domain documentation.		
Revision 0.2-21	Thu Jan 10 2013	Tom Wells
Upgraded build of the JBoss Enterprise Application Platform 6 Administration and Configuration Guide, for the 6.0.1 release. Includes updated topics, Bugzilla fixes, and new sections.		
Revision 0.1-1	Tue Dec 18 2012	Tom Wells
JBoss Enterprise Application Platform 6.0.1 Administration and Configuration Guide.		
Revision 0.0-18	Fri Dec 14 2012	Tom Wells
Bugzilla fixes, new topics, and structural changes for the JBoss Enterprise Application Platform 6 Administration and Configuration Guide.		
Revision 0.0-17	Tue Dec 11 2012	Tom Wells
Updated build of the JBoss Enterprise Application Platform 6.0.1 Administration and Configuration Guide.		
Revision 0.0-16	Mon Dec 03 2012	Tom Wells
Updated build of the JBoss Enterprise Application Platform 6.0.1 Administration and Configuration Guide.		
Revision 0.0-15	Fri Nov 30 2012	Tom Wells
Updated build of the JBoss Enterprise Application Platform 6.0.1 Administration and Configuration Guide.		
Revision 0.0-14	Tue Nov 27 2012	Tom Wells
Bugzilla fixes, new topics, and structural changes for the JBoss Enterprise Application Platform 6		

Administration and Configuration Guide.

Revision 0.0-13	Tue Nov 13 2012	Tom Wells
Bugzilla fixes, new topics, and structural changes for the JBoss Enterprise Application Platform 6 Administration and Configuration Guide.		
Revision 0.0-13	Tue Nov 13 2012	Tom Wells
Bugzilla fixes, new topics, and structural changes for the JBoss Enterprise Application Platform 6 Administration and Configuration Guide.		
Revision 0.0-12	Tue Oct 09 2012	Tom Wells
Release update. Bugzilla fixes, new topics, and structural changes for the JBoss Enterprise Application Platform 6 Administration and Configuration Guide.		
Revision 0.0-11	Wed Oct 03 2012	Tom Wells
Updated documentation. Includes domain mode content for Amazon EC2 AMIs, bugzilla issue fixes, and updates.		
Revision 0.0-10	Fri Sept 28 2012	Tom Wells
Bugzilla fixes and updates for the Administration and Configuration guide.		
Revision 0.0-9	Fri Sept 21 2012	Tom Wells
Bugzilla fixes for the Administration and Configuration guide.		
Revision 0.0-8	Wed Sept 12 2012	Tom Wells
Bugzilla fixes and updated documentation for the Administration and Configuration guide.		
Revision 0.0-7	Mon Sept 3 2012	Tom Wells
Bug fixes for the August asynchronous update to the Administration and Configuration guide.		
Revision 0.0-6	Fri Aug 31 2012	Tom Wells
August update of the Administration and Configuration guide for JBoss Enterprise Application Platform 6.		
Revision 0.0-5	Fri Aug 31 2012	Tom Wells
Updated edition of the Administration and Configuration guide for JBoss Enterprise Application Platform 6.0.0.		
Revision 0.0-4	Fri Aug 24 2012	Tom Wells
Bugzilla fixes for the release.		
Revision 0.0-3	Wed Aug 22 2012	Tom Wells
Bugzilla fixes and final Amazon EC2 content update.		
Revision 0.0-2	Mon Aug 13 2012	Tom Wells
Bugzilla fix update for the EAP 6.0.0 Administration and Configuration guide.		
Revision 0.0-1	Wed Jun 20 2012	Tom Wells
First edition of the JBoss Enterprise Application Platform 6 Administration and Configuration Guide.		