Red Hat JBoss Enterprise Application Platform 6.4

How To Configure Server Security

How to Configure Server Security

Last Updated: 2017-12-12

Red Hat JBoss Enterprise Application Platform 6.4 How To Configure Server Security

How to Configure Server Security

Abstract

The purpose of this document is to provide a practical guide to securing Red Hat JBoss Enterprise Application Platform 6. More specifically, this guide details how to secure all of the management interfaces on JBoss EAP 6. Before reading this guide, users should read through the Security Architecture document for Red Hat JBoss Enterprise Application Platform 6 and have a solid understanding of how JBoss EAP 6 handles security. This document also makes use of the JBoss EAP 6 CLI interface for performing configuration changes. For more information on using the CLI for both standalone JBoss EAP 6 instances as well as JBoss EAP 6 domains, please consult the The Management CLI section of the Administration and Configuration guide. When completing this document, readers should have a solid, working understanding of how to secure JBoss EAP 6.

Table of Contents

CHAPTER 1. OVERVIEW OF SECURITY

The basics of JBoss EAP 6 security as well as general security concepts are covered in the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide. Prior to reading this guide, it is important to understand the basic information covered in the Security Architecture guide around authentication, authorization, security realms, encryption, and SSL/TLS.

CHAPTER 2. SECURING THE SERVER AND ITS INTERFACES

2.1. BUILDING BLOCKS

2.1.1. Interfaces and Socket Bindings

JBoss EAP 6 utilizes its host's interfaces (e.g. inet-address, nic, etc) and ports for communication for both its web applications as well as its management interfaces. These interfaces and ports are defined and configured through the *interfaces* and *socket-binding-groups* settings in the JBoss EAP 6 configuration files (e.g. **`standalone.xml`**, **`domain.xml`**, **`host.xml`**, etc). For more information on how to define and configure *interfaces* and *socket-binding-groups*, consult the Interfaces and Socket Binding Groups sections of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide.

Example Interfaces

```
<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>
```

Example Socket Binding Group

```
<socket-binding-group name="standard-sockets" default-interface="public"
port-offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="management-native" interface="management"
port="${jboss.management.native.port:9999}"/>
  <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9443}"/>
  <socket-binding name="ajp" port="8009"/>
  <socket-binding name="http" port="8080"/>
  <socket-binding name="https" port="8443"/>
  <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>
```

2.1.2. Security Realms

JBoss EAP 6 uses security realms to define authentication and authorization mechanisms (e.g. local, LDAP, properties, etc) which can then be used by the management interfaces. For more background information on security realms, consult the *Security Realms* section of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide.

Example Security Realms

```
<security-realms>
  <security-realm name="ManagementRealm">
    <authentication>
      <local default-user="$local" skip-group-loading="true"/>
      <properties path="mgmt-users.properties" relative-
to="jboss.server.config.dir"/>
    </authentication>
    <authorization map-groups-to-roles="false">
      <properties path="mgmt-groups.properties" relative-
to="jboss.server.config.dir"/>
    </authorization>
  </security-realm>
  <security-realm name="ApplicationRealm">
    <authentication>
      <local default-user="$local" allowed-users="*" skip-group-
loading="true"/>
      <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>
```

NOTE

In addition to updating the existing security realms, JBoss EAP 6 also allows for new security realms to be created and used. New security realms may be created via the Management Console as well as invoking the following command from the Management CLI: */core-service=management/security-realm=NEW-REALM-NAME:add()*

2.1.3. Using Security Realms and Socket Bindings for Securing the Management Interfaces

JBoss EAP 6 defines two distinct management interfaces:

- http-interface
- native-interface

These interfaces are defined in the *<management-interfaces>* section of the JBoss EAP 6 configuration:

```
<management-interfaces>
  <native-interface security-realm="ManagementRealm">
    <socket-binding native="management-native"/>
  </native-interface>
  <http-interface security-realm="ManagementRealm">
```

```
    <socket-binding http="management-http"/>
  </http-interface>
</management-interfaces>
```

Notice that each interface specifies a *security-realm* and *socket-binding*. Updating the configuration for the specified security realm and socket binding allows for the management interfaces to be secured in different ways. In addition to being able to secure each of these interfaces via security realms and socket bindings, both of these interfaces also may be completely disabled, and users of these interfaces may be configured to have various roles and access rights. There are also a few topics in this guide, such as security auditing, secure passwords and JMX that overlap with other subsystems within JBoss EAP 6, but still relate to securing JBoss EAP 6.

2.2. HOW TO SECURE THE MANAGEMENT INTERFACES

The following sections show how to perform various operations related to securing the JBoss EAP 6 management interfaces and related subsystems.

NOTE

The below CLI commands were done assuming a standalone instance of JBoss EAP 6. For more details on using the CLI with JBoss EAP 6 domains, please consult *The Management CLI* section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Guide.

2.2.1. Configuring the Networking and Ports Used by Red Hat JBoss Enterprise Application Platform 6

Depending on the configuration of the host, JBoss EAP 6 may be configured to use various network interfaces and ports. This allows JBoss EAP 6 to work with different host, networking, and firewall requirements. For more information on the Networking and Ports used by JBoss EAP 6 as well as how to configure those settings, please see the *Network and Port Configuration* section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration guide.

2.2.2. Configure the Management Console for HTTPS

Configuring the JBoss EAP 6 management console for communication only via HTTPS provides increased security. All network traffic between the client (web browser) and management console is encrypted, which reduces the risk of security attacks such as a man-in-the-middle attack.

In this procedure unencrypted communication with the JBoss EAP 6 standalone instance or domain is disabled. This procedure applies to both standalone and domain mode configurations. For domain mode, prefix the Management CLI commands with the name of the host, for example: **`/host=master`**.

To configure the Management Console for HTTPS, the following steps are required:

1. Create a Keystore to Secure the Management Console
2. Ensure the Management Console Binds to HTTPS
3. Optional: Custom socket-binding Group
4. Create a New Security Realm
5. Configure Management Interface to Use the New Security Realm

6. Configure the Management Console to Use the Keystore

7. Restart the Red Hat JBoss Enterprise Application Platform 6 Instance

2.2.2.1. 1. Create a Keystore to Secure the Management Console

NOTE

This keystore must be in JKS format as the management console is not compatible with keystores in JCEKS format.

Use the following to generate a keystore, replacing the example values for the parameters (e.g. alias, keypass, keystore, storepass and dname) with the correct values for the environment.

NOTE

The parameter *validity* specifies for how many days the key is valid. A value of 730 equals two years.

Using the `keytool` command to generate a keystore from the terminal

```
keytool -genkeypair -alias appserver -storetype jks -keyalg RSA -keysize
2048 -keypass password1 -keystore
EAP_HOME/standalone/configuration/identity.jks -storepass password1 -dname
"CN=appserver,OU=Sales,O=Systems Inc,L=Raleigh,ST=NC,C=US" -validity 730 -
v
```

2.2.2.2. 2. Ensure the Management Console Binds to HTTPS

Standalone Mode

Ensure the management console binds to HTTPS for its interface by adding the *management-https* configuration and removing the *management-http* configuration.

Use the following CLI commands to bind the Management Console to HTTPS:

```
/core-service=management/management-interface=http-interface:write-
attribute( \
name=secure-socket-binding, value=management-https)
```

```
/core-service=management/management-interface= \
http-interface:undefine-attribute(name=socket-binding)
```

NOTE

At this point the JBoss EAP 6 log may display the following error message: *JBAS015103: A secure port has been specified for the HTTP interface but no SSL configuration in the realm.* This is to be expected because the SSL configuration is not yet completed.

Domain Mode

Change the socket element within the *management-interface* section by adding *secure-port* and removing port configuration.

Use the following commands to bind the Management Console to HTTPS:

```
/host=master/core-service=management/management-interface= \
http-interface:write-attribute(name=secure-port,value=9443)
```

```
/host=master/core-service=management/management-interface= \
http-interface:undefine-attribute(name=port)
```

NOTE

At this point the JBoss EAP 6 log may display the following error message: *JBAS015103: A secure port has been specified for the HTTP interface but no SSL configuration in the realm.* This is to be expected because the SSL configuration is not yet completed.

2.2.2.3. 3. Optional: Custom socket-binding Group

If a custom socket-binding group is being used, ensure the management-https binding is defined (it is present by default, bound to port 9443). Edit the master configuration file (e.g. **`standalone.xml`**) to match the following:

Example XML

```
<socket-binding-group name="standard-sockets" default-interface="public"
    port-offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="management-native" interface="management"
    port="${jboss.management.native.port:9999}"/>
  <socket-binding name="management-http" interface="management"
    port="${jboss.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management"
    port="${jboss.management.https.port:9443}"/>
...
```

2.2.2.4. 4. Create a New Security Realm

In this example, the new security realm using HTTPS (ManagementRealmHTTPS) will use a properties file named **`https-mgmt-users.properties`** located in the **`EAP_HOME/standalone/configuration/`** directory for storing usernames and passwords. Usernames and passwords will be added to the file later, but for now, simply create an empty file named *https-mgmt-users.properties* and save it to that location. The below example shows using the **`touch`** command from the terminal, but other mechanisms (such as using a text editor) could be used as well:

Example using the `touch` command from the terminal to create an empty file

```
touch EAP_HOME/standalone/configuration/https-mgmt-users.properties
```

Next, enter the following CLI commands to create a new security realm named ManagementRealmHTTPS:

```
/core-service=management/security-realm=ManagementRealmHTTPS/:add
```

```
/core-service=management/security-
realm=ManagementRealmHTTPS/authentication= \
properties/:add(path=https-mgmt-users.properties, \
relative-to=jboss.server.config.dir)
```

Now that the new properties file and realm have been created, users have to be added to that property file for use by the realm. This is accomplished via the **`add-user`** script available in the **`EAP_HOME/bin/`** directory. When running the **`add-user`**, both the properties file and realm need to be specified using the **`-up`** and **`-r`** options respectively. From there, the **`add-user`** script will interactively prompt for the username and password information to store in the **`https-mgmt-users.properties`** file.

```
EAP_HOME/bin/add-user.sh -up EAP_HOME/standalone/configuration/https-mgmt-
users.properties -r ManagementRealmHTTPS
...
Enter the details of the new user to add.
Using realm 'ManagementRealmHTTPS' as specified on the command line.
...
Username : httpUser
Password requirements are listed below. To modify these restrictions edit
the add-user.properties configuration file.
 - The password must not be one of the following restricted values {root,
admin, administrator}
 - The password must contain at least 8 characters, 1 alphabetic
character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
 - The password must be different from the username
...
Password :
Re-enter Password :
About to add user 'httpUser' for realm 'ManagementRealmHTTPS'
...
Is this correct yes/no? yes
..
Added user 'httpUser' to file 'EAP_HOME/configuration/https-mgmt-
users.properties'
...
Is this new user going to be used for one AS process to connect to another
AS process?
e.g. for a slave host controller connecting to the master or for a
Remoting connection for server to server EJB calls.
yes/no? no
```

IMPORTANT

When configuring security realms that use property files to store usernames and passwords, it is recommended that each realm use a distinct properties file that is not shared with another realm.

2.2.2.5. 5. Configure Management Interface to Use the New Security Realm

CLI Command for Using a New Security Realm

```
/core-service=management/management-interface=http-interface/:write-
attribute( \
name=security-realm,value=ManagementRealmHTTPS)
```

-

2.2.2.6. 6. Configure the Management Console to Use the Keystore

Use the below CLI command to configure the management console to use the keystore. For the parameters file, password and alias their values must be copied from the first step.

CLI Command for Adding a Keystore to a Security Realm

```
/core-service=management/security-realm=ManagementRealmHTTPS/server-
identity= \
ssl:add(keystore-path=identity.jks, \
keystore-relative-to=jboss.server.config.dir, \
keystore-password=password1, alias=appserver)
```

```
reload
```

NOTE

To update the keystore password, use the following CLI command:

```
/core-service=management/security-
realm=ManagementRealmHTTPS/server-identity=ssl:write-
attribute(name=keystore-password,value=newpassword)
```

2.2.2.7. 7. Restart the Red Hat JBoss Enterprise Application Platform 6 Instance

On restarting the server the log should contain the following, just before the text which states the number of services that are started. The management console is now listening on port 9443, which confirms that the procedure was successful.

```
14:53:14,720 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015962:
Http management interface listening on https://127.0.0.1:9443/management
14:53:14,721 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015952:
Admin console listening on https://127.0.0.1:9443
```

2.2.3. Setting up Distinct HTTP and HTTPS Management Interfaces

The Management Interface can listen on distinct interfaces for HTTP and HTTPS connections. One scenario for this is to listen for encrypted traffic on an external network, and use unencrypted traffic on an internal network.

The *secure-interface* attribute specifies the network interface on which the host’s socket for HTTPS management communication should be opened, if a different interface should be used from that specified by the *interface* attribute. If it is not specified then the *interface* specified by the interface attribute is used.

IMPORTANT

The *secure-interface* attribute has no effect if the *secure-port* attribute is not set.

Note that when the server listens for HTTP and HTTPS traffic on the same interface, HTTPS requests received by the HTTP listener are automatically redirected to the HTTPS port. When distinct interfaces are used for HTTP and HTTPS traffic, no redirection is performed when an HTTPS request is received by the HTTP listener.

Here is an example **EAP_HOME/domain/configuration/host.xml** configuration that sets the *secure-interface* attribute to listen for HTTPS traffic on a distinct interface from HTTP traffic:

Example XML

```
<host name="master" xmlns="urn:jboss:domain:3.0">
    <management>
        <security-realms>
            <security-realm name="ManagementRealm">
                <authentication>
                    <local default-user="$local" />
                    <properties path="mgmt-users.properties"
                      relative-to="jboss.domain.config.dir"/>
                </authentication>
            </security-realm>
        </security-realms>
        <management-interfaces>
            <native-interface security-realm="ManagementRealm">
                <socket interface="management"
                  port="${jboss.management.native.port:9999}"/>
            </native-interface>
            <http-interface security-realm="ManagementRealm">
                <socket interface="management"
                  port="${jboss.management.http.port:9990}"
                  secure-port="${jboss.management.https.port:9943}"
                  secure-interface="secure-management"/>
            </http-interface>
        </management-interfaces>
    </management>
    <domain-controller>
        <local/>
        <!-- Alternative remote domain controller configuration with a
host and port -->
        <!-- <remote host="${jboss.domain.master.address}"
port="${jboss.domain.master.port:9999}" security-realm="ManagementRealm"/>
-->
    </domain-controller>
    <interfaces>
        <interface name="management">
            <inet-address
value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="secure-management">
            <inet-address value="${jboss.bind.address:10.10.64.1}"/>
        </interface>
    </interfaces>
</host>
```

2.2.4. Disabling the HTTP Interfaces

In certain scenarios, such as managed domain or certain production deployments, administrators may wish to disable access to the management interfaces via HTTP or remove the HTTP management interface entirely.

The following may be used to read the current configuration of the HTTP interface:

CLI Command for Reading the Current HTTP Interface Configuration

```
/core-service=management/management-interface=http-interface/:read-
resource( \
recursive=true, proxies=false, include-runtime=false, include-
defaults=true)
```

Result

```
{
    "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 entirely:

CLI Command for Removing the HTTP Interface Entirely

```
/core-service=management/management-interface=http-interface/:remove
```

Once the HTTP interface has been removed, it can be enabled again by re-creating it:

CLI Command for Re-Creating the HTTP Interface:

```
/core-service=management/management-interface=http-interface:add( \
console-enabled=true, interface=management, \
port="${jboss.management.http.port:9990}", \
security-realm=ManagementRealm)
```

Disabling Just the Management Console

Other clients, such as JBoss Operations Network, also operate using the HTTP interface for managing JBoss EAP 6. In order to continue using these services, just the web-based Management Console itself may be disabled. This is accomplished by setting the *console-enabled* attribute to *false*:

CLI Command for Disabling the Web-Based Management Console

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

2.2.5. Setting up 2-Way SSL/TLS for the Management Interfaces

2-way SSL/TLS authentication, also known as *client authentication*, authenticates both the client and the server using SSL/TLS certificates. This differs from the Configure the Management Console for HTTPS section in that both the client and server each have a certificate. This provides assurance that not only is the server who it says it is, but the client is also who it says it is.

In this section the following conventions are used:

HOST1

The JBoss server hostname. For example; jboss.redhat.com

HOST2

A suitable name for the client. For example: myclient. Note this is not necessarily an actual hostname.

CA_HOST1

The DN (distinguished name) to use for the HOST1 certificate. For example cn=jboss,dc=redhat,dc=com.

CA_HOST2

The DN (distinguished name) to use for the HOST2 certificate. For example cn=myclient,dc=redhat,dc=com.

To setup the 2-way SSL/TLS for the Management Interfaces, the following steps are required:

1. Generate the stores
2. Export the certificates
3. Import the certificates into the opposing trust stores
4. Define a CertificateRealm
5. Change the security-realm of the native-interface to the new Certificate Realm
6. Add the SSL configuration for the CLI

PREREQUISITES

If a password vault is used to store the keystore and truststore passwords (recommended), the password vault should already be created. For more information on the password vault, please see the Password Vault section as well as the *Password Vault System* section of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide.

1. Generate the stores:

```
keytool -genkeypair -alias HOST1_alias -keyalg RSA -keysize 1024 -validity
365 -keystore HOST1.keystore.jks -dname "CA_HOST1" -keypass secret -
storepass secret
```

```
keytool -genkeypair -alias HOST2_alias -keyalg RSA -keysize 1024 -validity
365 -keystore HOST2.keystore.jks -dname "CA_HOST2" -keypass secret -
storepass secret
```

2. Export the certificates:

```
keytool -exportcert  -keystore HOST1.keystore.jks -alias HOST1_alias -
keypass secret -storepass secret -file HOST1.cer
```

```
keytool -exportcert  -keystore HOST2.keystore.jks -alias HOST2_alias -
keypass secret -storepass secret -file HOST2.cer
```

3. Import the certificates into the opposing trust stores:

```
keytool -importcert -keystore HOST1.truststore.jks -storepass secret -alias
HOST2_alias -trustcacerts -file HOST2.cer
```

```
keytool -importcert -keystore HOST2.truststore.jks -storepass secret -alias
HOST1_alias -trustcacerts -file HOST1.cer
```

4. Define a CertificateRealm

Define a CertificateRealm in the configuration for the installation (**`host.xml`** or **`standalone.xml`**) and point the interface to it. This can be done using the following commands:

```
/core-service=management/security-realm=CertificateRealm:add()
```

```
/core-service=management/security-realm=CertificateRealm/server-identity=
\
ssl:add(keystore-path=/path/to/HOST1.keystore.jks, keystore-
password=secret, \
alias=HOST1_alias)
```

```
/core-service=management/security-realm=CertificateRealm/authentication= \
truststore:add(keystore-path=/path/to/HOST1.truststore.jks, \
keystore-password=secret)
```

5. Change the security-realm of the native-interface to the new Certificate Realm.

```
/core-service=management/management-interface= \
native-interface:write-attribute(name=security-
realm,value=CertificateRealm)
```

6. Add the SSL configuration for the CLI,

Add the SSL configuration for the CLI, which uses **`EAP_HOME/bin/jboss-cli.xml`** as a settings file. The passwords for the truststore and keystore can be stored in one of two ways:

- Password Vault (Recommended)
- Plain Text

6A. To store the keystore and truststore passwords in a password vault:

Edit **`EAP_HOME/bin/jboss-cli.xml`** and add the SSL/TLS configuration (using the appropriate values for the variables). Also add the vault configuration, replacing each value with those of the vault being used.

Example jboss-cli.xml XML

```
<ssl>
  <vault>
    <vault-option name="KEYSTORE_URL" value="path-
to/vault/vault.keystore"/>
    <vault-option name="KEYSTORE_PASSWORD" value="MASK-5WNXs8oEbrs"/>
    <vault-option name="KEYSTORE_ALIAS" value="vault"/>
    <vault-option name="SALT" value="12345678"/>
    <vault-option name="ITERATION_COUNT" value="50"/>
    <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
  </vault>
  <alias>HOST2_alias</alias>
  <key-store>/path/to/HOST2.keystore.jks</key-store>
  <key-store-password>VAULT::VB::cli_pass::1</key-store-password>
  <key-password>VAULT::VB::cli_pass::1</key-password>
  <trust-store>/path/to/HOST2.truststore.jks</trust-store>
  <trust-store-password>VAULT::VB::cli_pass::1</trust-store-password>
  <modify-trust-store>true</modify-trust-store>
</ssl>
```

6B. To store the keystore and truststore passwords in plain text:

Edit **`EAP_HOME/bin/jboss-cli.xml`** and add the SSL/TLS configuration (using the appropriate values for the variables):

Example jboss-cli.xml XML

```
<ssl>
  <alias>HOST2_alias</alias>
  <key-store>/path/to/HOST2.keystore.jks</key-store>
  <key-store-password>secret</key-store-password>
  <trust-store>/path/to/HOST2.truststore.jks</trust-store>
  <trust-store-password>secret</trust-store-password>
  <modify-trust-store>true</modify-trust-store>
</ssl>
```

2.2.6. Setting up an SSL/TLS Connector

In addition to supporting HTTPS and 2-way SSL/TLS for the management interfaces, JBoss EAP 6 also enables SSL/TLS connectors to be setup for use by security domains.

IMPORTANT

As a prerequisite, an SSL/TLS Encryption Key and Certificate should be created and place in an accessible directory. Additionally, relavant information (e.g. keystore aliases and passwords, desired cypher suites, etc) should also be accessable. For examples on generating SSL/TLS Keys and Certificates, please see the first two steps in the Setting up 2-Way SSL/TLS for the Management Interfaces section. For more information about the the SSL/TLS connector (including cypher suites) please see the SSL Connector Reference section.

Add a HTTPS connector in the web subsystem

Create a secure connector, named *HTTPS*, which uses the https scheme, the https socket binding (which defaults to 8443), and is set to be secure.

```
/subsystem=web/connector=HTTPS/:add(socket-binding=https,scheme=https, \
protocol=HTTP/1.1,secure=true)
```

Configure the SSL certificate, substituting the correct values for the example ones. This example assumes that the keystore is copied to the server configuration directory, which is **`EAP_HOME/standalone/configuration/`** for a standalone JBoss EAP 6 instance.

```
/subsystem=web/connector=HTTPS/ssl=configuration:add( \
name=https,certificate-key-file="${jboss.server.config.dir}/keystore.jks",
\
password=SECRET, key-alias=KEY_ALIAS, cipher-suite=CIPHERS)
```

Set the protocol to *TLSv1*.

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=protocol,value=TLSv1)
```

Deploy an application.

IMPORTANT

The JBoss EAP 6 instance must be restarted for these changes to take effect.

2.2.7. SSL Connector Reference

JBoss Web connectors may include the following SSL configuration attributes.

2.2.7.1. name

The display name of the SSL connector. Attribute name is read-only.

2.2.7.2. verify-client

The possible values of verify-client differ, based upon whether the HTTP/HTTPS connector is used, or the native APR connector is used.

HTTP/HTTPS Connector

Possible values are *true*, *false*, or *want*. Set to *true* to require a valid certificate chain from the client before accepting a connection. Set to *want* if to 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.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-
attribute(name=verify-client,value=want)
```

Native APR Connector

Possible values are *optional*, *require*, *optionalNoCA*, and *none* (or any other string, which will have the same effect as *none*). These values determine whether a certification is optional, required, optional without a Certificate Authority, or not required at all. The default is none, meaning the client will not have the opportunity to submit a certificate.

Example CLI

```
/subsystem=web/connector=APR/ssl=configuration/:write-attribute( \
name=verify-client,value=require)
```

2.2.7.3. 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.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=verify-depth,value=10)
```

2.2.7.4. 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 EAP 6. If the keystoreType does not use a file, set the parameter to an empty string.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=certificate-key-file, \
value=../domain/configuration/server.keystore)
```

2.2.7.5. certificate-file

If OpenSSL encryption is used, set the value of this parameter to the path to the file containing the server certificate.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=certificate-file,value=server.crt)
```

2.2.7.6. password

The password for both the truststore and keystore. In the following example, replace PASSWORD with the actual password.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=password,value=PASSWORD)
```

2.2.7.7. protocol

The version of the SSL protocol to use. Supported values depend on the underlying SSL implementation (whether JSSE or OpenSSL). Refer to the Java SSE Documentation.

A comma separated combination of protocols can also be specified. For example, TLSv1, TLSv1.1,TLSv1.2.

WARNING

Red Hat recommends that SSL is explicitly disabled in favor of TLSv1.1 or TLSv1.2 in all affected packages.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=protocol,value=ALL)
```

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=protocol,value="TLSv1, TLSv1.1,TLSv1.2")
```

2.2.7.8. cipher-suite

A list of the encryption ciphers which are allowed. For JSSE syntax, it must be a comma-separated list. For OpenSSL syntax, it must be a colon-separated list. Ensure that only one syntax is used.

The default is *HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5*.

The example only lists two possible ciphers, but real-world examples will likely use more.

IMPORTANT

Using weak ciphers is a significant security risk. See http://www.nist.gov/manuscript-publication-search.cfm?pub_id=915295 for NIST recommendations on cipher suites.

For a list of available OpenSSL ciphers, see https://www.openssl.org/docs/manmaster/apps/ciphers.html#CIPHER-STRINGS. Note that the following are not supported:

- *@SECLEVEL*
- *SUITEB128*
- *SUITEB128ONLY*
- *SUITEB192*

For a list of the standard JSSE ciphers, see http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Cipher.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=cipher-suite, \
value="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA")
```

2.2.7.9. key-alias

The alias used to for the server certificate in the keystore. In the following example, replace KEY_ALIAS with the certificate's alias.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=key-alias,value=KEY_ALIAS)
```

2.2.7.10. truststore-type

The type of the truststore. Various types of truststores are available, including PKCS12 and Java's standard JKS.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=truststore-type,value=jks)
```

2.2.7.11. keystore-type

The type of the keystore, Various types of keystores are available, including PKCS12 and Java's standard JKS.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=keystore-type,value=jks)
```

2.2.7.12. 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.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=ca-certificate-file,value=ca.crt)
```

2.2.7.13. ca-certificate-password

The Certificate password for the ca-certificate-file. In the following example, replace the MASKED_PASSWORD with the masked password.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=ca-certificate-password, value=MASKED_PASSWORD)
```

2.2.7.14. ca-revocation-url

A file or URL which contains the revocation list. It refers to the crlFile for JSSE or the SSLCARevocationFile for SSL.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=ca-revocation-url,value=ca.crl)
```

2.2.7.15. session-cache-size

The size of the SSLSession cache. This attribute applies only to JSSE connectors. The default is 0, which specifies an unlimited cache size.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=session-cache-size,value=100)
```

2.2.7.16. session-timeout

The number of seconds before a cached SSLSession expires. This attribute applies only to JSSE connectors. The default is 86400 seconds, which is 24 hours.

Example CLI

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=session-timeout,value=43200)
```

2.2.8. Disabling Remote Access to JMX

Remote access to the JMX subsystem allows for JDK and application management operations to be triggered remotely. To disable remote access to JMX in JBoss EAP 6, one of the following two things must be done:

1. Remove the remoting connector in the JMX subsystem

2. Remove the entire JMX subsystem

Removing the remoting connector

```
/subsystem=jmx/remoting-connector=jmx/:remove
```

Removing the entire JMX subsytem

```
/subsystem=jmx/:remove
```

For more information on JMX, please see the JMX section of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide

2.2.9. Using JAAS for Securing the Management Interfaces

JAAS is a declarative security API used by JBoss EAP 6 to manage security. For more details and background regarding JAAS and declarative security, see the Declarative Security and JAAS section of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide.

NOTE

When JBoss EAP 6 instances are configured to run in **ADMIN_ONLY** mode, using JAAS to secure the management interfaces is not supported. For more information on **ADMIN_ONLY** mode, please see section *Reference of Switches and Arguments to pass at Server Runtime* of the Administration and Configuration Guide.

To use JAAS to authenticate to the management interfaces, the following steps must be performed:

1. Create a security domain
2. Create a security realm with JAAS authentication
3. Update the Management Interfaces to use new security realm
4. *Optional* - Assign group membership

1. Creating a security domain

In this example, a security domain is created with the UserRoles login module, but other login modules may be used as well:

```
/subsystem=security/security-domain=UsersLMDomain:add( \
cache-type=default)
```

```
/subsystem=security/security-
domain=UsersLMDomain/authentication=classic:add
```

```
/subsystem=security/security-domain=UsersLMDomain/authentication= \
classic/login-module=UsersRoles:add(code=UsersRoles, flag=required, \
module-options=[("usersProperties"=>"users.properties"), \
("rolesProperties"=>"roles.properties")])
```

2. Create a security realm with JAAS authentication

To create a security realm with JAAS Authentication:

```
/core-service=management/security-realm=SecurityDomainAuthnRealm:add
```

```
/core-service=management/security-realm= \
SecurityDomainAuthnRealm/authentication=jaas:add(name=UsersLMDomain)
```

3. Update the Management Interfaces to use new security realm

To update the *http-interface* to use the new realm:

```
/core-service=management/management-interface=http-interface/:write-
attribute( \
name=security-realm,value=SecurityDomainAuthnRealm)
```

To update the *native-interface* to use the new realm:

```
/core-service=management/management-interface=native-interface/:write-
attribute( \
name=security-realm,value=SecurityDomainAuthnRealm)
```

4. *Optional* - Assign group membership

The attribute assign-groups determines whether loaded user membership information from the security domain is used for group assignment in the security realm. When set to true this group assignment is used for Role-Based Access Control (RBAC).

To configure the assign-groups attribute:

```
/core-service=management/security-
realm=SecurityDomainAuthnRealm/authentication= \
jaas:write-attribute(name=assign-groups,value=true)
```

2.2.10. Silent Authentication

The default installation of JBoss EAP 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 instances as well as domains.

IMPORTANT

The removal of the local element should only be done if the impact on the JBoss EAP 6 instance and its configuration is fully understood.

To remove silent authentication from a realm:

```
/core-service=management/security-
realm=REALM_NAME/authentication=local:remove
```

2.3. 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 or the management interfaces. Auditing mechanisms are configured as part of a security domain or management interface.

Auditing uses provider modules. Both included provider modules as well as custom implementations may be used.

2.3.1. Configure Security Auditing for the Management Interfaces

For more information on configuring auditing for the management interfaces, refer to the Management Interface Audit Logging section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide.

2.3.2. Configure Security Auditing for Security Domains

To configure security auditing settings for a security domain, the following steps must be performed from the management console:

1. Open the security domain’s detailed view.
2. Navigate to the Auditing subsystem configuration.
3. Add a provider module.
4. Verify the module is working
5. Optional: Add, edit, or remove module options.

1. Open the security domain’s detailed view.

- Click *Configuration* at the top of the screen.
- In a managed domain, select a profile to modify from the *Profile* selection box at the top left.
- Expand the *Security* menu and select *Security Domains*.
- Click *View* for the security domain to edit.

2. Navigate to the Auditing subsystem configuration.

Select the Audit tab at the top of the screen.

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 *Add* and fill in the *Code* section with the classname of the provider module.

4. Verify the module is working

The goal of an audit module is to provide a way to monitor the events in the security subsystem. This monitoring can be done by means of writing to a log file, email notifications or any other measurable auditing mechanism.

For example, JBoss EAP 6 includes the *org.jboss.security.audit.providers.LogAuditProvider* module by default. If enabled following the steps above , this audit module writes security notifications to a **`audit.log`** file in the log subfolder within the **`EAP_HOME`** directory.

To verify if the steps above have worked in the context of the *org.jboss.security.audit.providers.LogAuditProvider*, perform an action that is likely to trigger a notification and then check the audit log file.

5.Optional: Add, edit, or remove module options.

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 *Add*, and provide the key and value for the option.

To edit an option that already exists, click *Remove* to remove it, and click *Add* to add it again with the correct options.

CHAPTER 3. SECURING USERS OF THE SERVER AND ITS MANAGEMENT INTERFACES

In addition to understanding how to secure the various interfaces of JBoss EAP 6, its also important to understand how to secure the users that access those interfaces.

3.1. USER AUTHENTICATION

3.1.1. Default User Configuration

All management interfaces in JBoss EAP 6 are secured by default and users can access them in two different ways: local interfaces and remote interfaces. The basics of both of these authentication mechanisms are covered in the Default Security and Red Hat JBoss Enterprise Application Platform Out of the Box sections of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide. By default, access to these interfaces is configured in the *Management Realm* security realm. Initially, the local interface is enabled and requires access to the host machine running the JBoss EAP 6 instance. Remote access is also enabled and is configured to use a file-based identity store. By default it uses **`mgmt-users.properties`** file to store usernames and passwords, and **`mgmt-groups.properties`** to store user group information.

User information is added to these files by using the included **`adduser`** script located in the **`EAP_HOME/bin/`** directory.

To add a user via the `adduser` script:

1. Run the **`add-user.sh`** or **`add-user.bat`** command.

2. Choose whether to add a Management User or Application User.

3. Choose the realm the user will be added to. By default, the only available realms are ManagementRealm and ApplicationRealm. If a custom realm has been added, its name can be manually entered instead.

4. Type the desired username, password, and optional roles when prompted. The changes are written to each of the properties files for the security realm.

3.1.2. Adding Authentication via LDAP

JBoss EAP 6 also supports using LDAP authentication for securing the management interfaces. The basics of LDAP and how it works with JBoss EAP are covered in the LDAP, Using LDAP with the Management Interfaces, and Using LDAP with the ManagementRealm sections of the the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide. For more specifics on how to secure the management interfaces using LDAP authentication, see the *Securing the Management Interfaces with LDAP* section of the How to Configure Identity Management guide.

3.2. SECURE PASSWORDS

3.2.1. Password Vault

Configuration of JBoss EAP 6 and associated applications requires potentially sensitive information, such as usernames and passwords. Instead of storing the password as plain text in configuration files, the Password Vault feature can be used to mask the password information and store it in an encrypted keystore. Once the password is stored, references can be included in Management CLI commands or applications deployed to JBoss EAP 6.

The Password Vault uses the Java Keystore as its storage mechanism. Password Vault consists of two parts: storage and key storage. Java Keystore is used to store the key, which is used to encrypt or decrypt sensitive strings in Vault storage.

To setup and use a Password Vault requires the following steps:

1. Setup a Java Keystore to Store Key for Password Encryption
2. Initialize the Password Vault
3. Configure Red Hat JBoss Enterprise Application Platform 6 to use the Password Vault
4. Store a Sensitive String in the Password Vault
5. Use an Encrypted Sensitive String in Configuration

3.2.1.1. 1. Setup a Java Keystore to Store Key for Password Encryption

NOTE

The keytool utility, provided by the Java Runtime Environment (JRE), is utilized for this steps. Locate the path for the file, which on Red Hat Enterprise Linux is **`/usr/bin/keytool`**.

WARNING

JCEKS keystore implementations differ between Java vendors so the keystore must be generated using the keytool utility from the same vendor as the JDK used. Using a keystore generated by the keytool from one vendor's JDK in a JBoss EAP 6 instance running on a JDK from a different vendor results in the following exception: **java.io.IOException: com.sun.crypto.provider.SealedObjectForKeyProtector**

To setup the keystore, perform the following steps:

1. Create a directory to store the keystore and other encrypted information.
2. Determine the parameters to use with keytool utility.
3. Run the keytool command

1. Create a directory to store the keystore and other encrypted information

Create a directory to store the keystore and other important information. The rest of this procedure assumes that the directory is **`EAP_HOME/vault/`**. Since this directory will contain sensitive information it should be accessible to only limited users. At a minimum the user account under which JBoss EAP 6 is running requires read-write access.

2. Determine the parameters to use with keytool utility.

Decide on values for the following parameters:

alias

The alias is a unique identifier for the vault or other data stored in the keystore. Aliases are case-insensitive.

storetype

The storetype specifies the keystore type. The value *jceks* is recommended.

keyalg

The algorithm to use for encryption. Use the documentation for the JRE and operating system to see which other choices are available.

keysize

The size of an encryption key impacts how difficult it is to decrypt through brute force. For information on appropriate values, see the documentation distributed with the keytool utility.

storepass

The value of storepass is the password that 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. If this parameter is omitted, the keytool utility will prompt for it to be entered after the command has been executed

keypass

The value of keypass is the password used to access the specific key and must match the value of the storepass parameter.

validity

The value of validity is the period (in days) for which the key will be valid.

keystore

The value of keystore is the filepath and filename in which the keystore's values are to be stored. The keystore file is created when data is first added to it. Ensure the correct file path separator is used: / (forward slash) for Red Hat Enterprise Linux and similar operating systems, \ (backslash) for Microsoft Windows Server.

The keytool utility has many other options. See the documentation for the JRE or the operating system for more details.

3. Run the keytool command

```
$ keytool -genseckey -alias vault -storetype jceks -keyalg AES -keysize
128 -storepass vault22 -keypass vault22 -validity 730 -keystore
EAP_HOME/vault/vault.keystore
```

This results in a keystore that has been created in the file **`EAP_HOME/vault/vault.keystore`**. It stores a single key, with the alias vault, which will be used to store encrypted strings, such as passwords, for JBoss EAP 6.

3.2.1.2. 2. Initialize the Password Vault

The Password Vault can be initialized either interactively, where you are prompted for each parameter's value, or non-interactively, where all parameters' values are provided on the commmand line. Each method gives the same result, so either may be used.

The following parameters will be needed:

Keystore URL (KEYSTORE_URL)

The file system path or URI of the keystore file. The examples use **`EAP_HOME/vault/vault.keystore`**.

Keystore password (KEYSTORE_PASSWORD)

The password used to access the keystore.

Salt (SALT)

The salt value is a random string of eight characters used, together with the iteration count, to encrypt the content of the keystore.

Keystore Alias (KEYSTORE_ALIAS)

The alias by which the keystore is known.

Iteration Count (ITERATION_COUNT)

The number of times the encryption algorithm is run.

Directory to store encrypted files (ENC_FILE_DIR)

The path in which the encrypted files are to be stored. This is typically the directory containing the password vault. It is convenient but not mandatory to store all of your encrypted information in the same place as the key store. This directory should be only accessible to limited users. At a minimum the user account under which JBoss EAP 6 is running requires read-write access. The keystore should be located in the directory used for Step 1. Note that the trailing backslash or forward slash on the directory name is required. Ensure the correct file path separator is used: / (forward slash) for Red Hat Enterprise Linux and similar operating systems, \ (backslash) for Microsoft Windows Server.

Vault Block (VAULT_BLOCK)

The name to be given to this block in the password vault.

Attribute (ATTRIBUTE)

The name to be given to the attribute being stored.

Security Attribute (SEC-ATTR)

The password which is being stored in the password vault.

Running the vault command non-interactively

To run the password vault command non-interactively, the *vault* script (located in **`EAP_HOME/bin/`**) can be invoked with parameters for the relavant information:

```
vault.sh --keystore KEYSTORE_URL --keystore-password KEYSTORE_PASSWORD --
alias KEYSTORE_ALIAS --vault-block VAULT_BLOCK --attribute ATTRIBUTE --
sec-attr SEC-ATTR --enc-dir ENC_FILE_DIR --iteration ITERATION_COUNT --
salt SALT
```

Example

```
vault.sh --keystore EAP_HOME/vault/vault.keystore --keystore-password
vault22 --alias vault --vault-block vb --attribute password --sec-attr
0penS3sam3 --enc-dir EAP_HOME/vault/ --iteration 120 --salt 1234abcd
```

Output

```
 =========================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 ==========================================================================
Oct 17, 2014 2:23:43 PM org.picketbox.plugins.vault.PicketBoxSecurityVault
init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Secured attribute value has been stored in vault.
Please make note of the following:
********************************************
Vault Block:vb
Attribute Name:password
Configuration should be done as follows:
VAULT::vb::password::1
********************************************
Vault Configuration in AS7 config file:
********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL"
value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
********************************************
```

Running the vault command interactively

To run the password vault command interactively, the following steps are required:

1. Launch the Password Vault command interactively
2. Complete the prompted parameters
3. Make a note of the masked password information
4. Exit the interactive console

1. Launch the Password Vault command interactively

Run **`EAP_HOME/bin/vault.sh`** (on Red Hat Enterprise Linux and similar operating systems) or **`EAP_HOME\bin\vault.bat`** (on Microsoft Windows Server). Start a new interactive session by typing *0* (zero).

2. Complete the prompted parameters

Follow the prompts to input the required parameters.

3. Make a note of the masked password information

The masked password, salt, and iteration count are printed to standard output. Make a note of them in a secure location. They are required to add entries to the Password Vault. Access to the keystore file and these values could allow an attacker access to obtain access to sensitive information in the Password Vault.

4. Exit the interactive console

Type *2* (two) to exit the interactive console.

Example Input and Output

```
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:EAP_HOME/vault/
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password: vault22
Enter Keystore password again: vault22
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Oct 17, 2014 12:58:11 PM
org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Vault Configuration in AS7 config file:
********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL"
value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
********************************************
Vault is initialized and ready for use
Handshake with Vault complete
```

The keystore password has been masked for use in configuration files and deployments. In addition, the vault is initialized and ready to use.

3.2.1.3. 3. Configure Red Hat JBoss Enterprise Application Platform 6 to use the Password Vault

Before passwords and other sensitive attributes can be masked and used in configuration files, JBoss EAP 6 must be made aware of the password vault which stores and decrypts them.

The following command can be used to configure JBoss EAP 6 to use the password vault:

```
/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")])
```

Example CLI Command

```
/core-service=vault:add( \
vault-options=[ \
("KEYSTORE_URL" => "EAP_HOME/vault/vault.keystore"), \
("KEYSTORE_PASSWORD" => "MASK-5dOaAVafCSd"), \
("KEYSTORE_ALIAS" => "vault"), \
("SALT" => "1234abcd"), \
("ITERATION_COUNT" => "120"), \
("ENC_FILE_DIR" => "EAP_HOME/vault/")])
```

NOTE

If Microsoft Windows Server is being used, use two backslashes (\\) in the file path instead using one. For example, **`C:\\data\\vault\\vault.keystore`**. This is because a single backslash character (\) is used for character escaping.

JBoss EAP 6 is now configured to decrypt masked strings stored in the Password Vault.

3.2.1.4. 4. Store a Sensitive String in the Password Vault

Including passwords and other sensitive strings in plaintext configuration files is a security risk. Store these strings instead in the Password Vault for improved security, where they can then be referenced in configuration files, management CLI commands and applications in their masked form.

Sensitive strings can be stored in the Password Vault either interactively, where the tool prompts for each parameter's value, or non-interactively, where all the parameters' values are provided on the commmand line. Each method gives the same result, so either may be used. Both of these methods are invoked using the *vault* script.

Store a Sensitive String Non-interactively

To run the password vault command non-interactively, the *vault* script (located in **`EAP_HOME/bin/`**) can be invoked with parameters for the relavant information:

```
EAP_HOME/bin/vault.sh --keystore KEYSTORE_URL --keystore-password
KEYSTORE_PASSWORD --alias KEYSTORE_ALIAS --vault-block VAULT_BLOCK --
attribute ATTRIBUTE --sec-attr SEC-ATTR --enc-dir ENC_FILE_DIR --iteration
ITERATION_COUNT --salt SALT
```

NOTE

The keystore password must be given in plaintext form, not masked form.

Example

```
EAP_HOME/bin/vault.sh --keystore EAP_HOME/vault/vault.keystore --keystore-
password vault22 --alias vault --vault-block vb --attribute password --
sec-attr 0penS3sam3 --enc-dir EAP_HOME/vault/ --iteration 120 --salt
```

```
1234abcd
```

Output

```
 =========================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 =========================================================================
Oct 22, 2014 9:24:43 AM org.picketbox.plugins.vault.PicketBoxSecurityVault
init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Secured attribute value has been stored in vault.
Please make note of the following:
********************************************
Vault Block:vb
Attribute Name:password
Configuration should be done as follows:
VAULT::vb::password::1
********************************************
Vault Configuration in AS7 config file:
********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL"
value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="vault22"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/vault/"/>
</vault><management> ...
********************************************
```

After invoking the *vault* script, a message prints to standard output, showing the vault block, attribute name, masked string, and advice about using the string in your configuration. Make note of this information in a secure location. An extract of sample output is as follows:

```
Vault Block:vb
Attribute Name:password
Configuration should be done as follows:
VAULT::vb::password::1
```

Store a Sensitive String Interactively

To run the password vault command interactively, the following steps are required:

1. Launch the Password Vault command interactively.
2. Complete the prompted parameters.
3. Complete the prompted parameters about the sensitive string

4. Make note of the information about the masked string

5. Exit the interactive console.

1. Launch the Password Vault command interactively.

Launch the operating system's command line interface and run **`EAP_HOME/bin/vault.sh`** (on Red Hat Enterprise Linux and similar operating systems) or **`EAP_HOME\bin\vault.bat`** (on Microsoft Windows Server). Start a new interactive session by typing *0* (zero).

2. Complete the prompted parameters.

Follow the prompts to input the required parameters. These values must match those provided when the Password Vault was created.

NOTE

The keystore password must be given in plaintext form, not masked form.

3. Complete the prompted parameters about the sensitive string

Enter *0* (zero) to start storing the sensitive string. Follow the prompts to input the required parameters.

4. Make note of the information about the masked string

A message prints to standard output, showing the vault block, attribute name, masked string, and advice about using the string in the configuration. Make note of this information in a secure location. An extract of sample output is as follows:

```
Vault Block:ds_Example1
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_Example1::password::1
```

5. Exit the interactive console.

Type *2* (two) to exit the interactive console.

Example Input and Output

```
  =========================================================================
   JBoss Vault
   JBOSS_HOME: EAP_HOME/jboss-eap-6.4
   JAVA: java
  =========================================================================
  **********************************
  ****  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:EAP_HOME/vault/
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
```

```
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Oct 21, 2014 11:20:49 AM
org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Vault Configuration in AS7 config file:
 ********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL"
value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
 ********************************************
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::  0: Store a secured attribute  1: Check whether a
secured attribute exists  2: Remove secured attribute  3: Exit
0
Task: Store a secured attribute
Please enter secured attribute value (such as password):
Please enter secured attribute value (such as password) again:
Values match
Enter Vault Block:ds_Example1
Enter Attribute Name:password
Secured attribute value has been stored in vault.
Please make note of the following:
 ********************************************
Vault Block:ds_Example1
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_Example1::password::1
 ********************************************
Please enter a Digit::  0: Store a secured attribute  1: Check whether a
secured attribute exists  2: Remove secured attribute  3: Exit
```

3.2.1.5. 5. Use an Encrypted Sensitive String in Configuration

Any sensitive string which have been encrypted can be used in a configuration file or Management CLI command in its masked form, providing expressions are allowed.

To confirm if expressions are allowed within a particular subsystem, run the following Management CLI command against that subsystem:

```
/subsystem=SUBSYSTEM:read-resource-description(recursive=true)
```

From the output of running this command, look for the value of the *expressions-allowed* parameter. If this is true, then expressions can be used within the configuration of this subsystem.

Use the following syntax to replace any plaintext string with the masked form.

```
${VAULT::VAULT_BLOCK::ATTRIBUTE_NAME::MASKED_STRING}
```

Example - Datasource Definition Using a Password in Masked Form

```
...
  <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::1}</password>
        </security>
      </datasource>
      <drivers>
         <driver name="h2" module="com.h2database.h2">
            <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-
datasource-class>
         </driver>
      </drivers>
    </datasources>
  </subsystem>
...
```

3.2.1.6. Use an Encrypted Sensitive String in an Application

Encrypted strings stored in the Password Vault can be used in an application's source code. The below example is an extract of a servlet's source code, illustrating the use of a masked password in a datasource definition, instead of the plaintext password. The plaintext version is commented out so that you can see the difference.

Servlet Using a Vaulted Password

```
@DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "VAULT::DS::thePass::1",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
)
/*old (plaintext) definition
@DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "sa",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
)*/
```

3.2.1.7. Check if a Sensitive String is in the Password Vault

Before attempting to store or use a sensitive string in the Password Vault it can be useful to first confirm if it is already stored.

This check can be done either interactively, where the user is prompted for each parameter’s value, or non-interactively, where all parameters' values are provided on the commmand line. Each method gives the same result, so either may be used. Both of these methods are invoked using the *vault* script.

Check For a Sensitive String Non-Interactively

Use this method to provide all parameters' values at once. For a description of all parameters, see Initialize the Password Vault.

To run the password vault command non-interactively, the *vault* script (located in **EAP_HOME/bin/**) can be invoked with parameters for the relavant information:

```
EAP_HOME/bin/vault.sh --keystore KEYSTORE_URL --keystore-password
KEYSTORE_PASSWORD --alias KEYSTORE_ALIAS --check-sec-attr --vault-block
VAULT_BLOCK --attribute ATTRIBUTE --enc-dir ENC_FILE_DIR --iteration
ITERATION_COUNT --salt SALT
```

Substitute the placeholder values with the actual values. The values for parameters KEYSTORE_URL, KEYSTORE_PASSWORD and KEYSTORE_ALIAS must match those provided when the Password Vault was created.

NOTE

The keystore password must be given in plaintext form, not masked form.

If the sensitive string is stored in the vault block specified, the following message will be displayed:

```
Password already exists.
```

If the value is not stored in the specified block, the following message will be displayed:

```
Password doesn't exist.
```

Check For a Sensitive String Interactively

To run the password vault command interactively, the following steps are required:

1. Launch the Password Vault command interactively
2. Complete the prompted parameters

1. Launch the Password Vault command interactively

Run **EAP_HOME/bin/vault.sh** (on Red Hat Enterprise Linux and similar operating systems) or **EAP_HOME\bin\vault.bat** (on Microsoft Windows Server). Start a new interactive session by typing *0* (zero).

2. Complete the Prompted Parameters

Follow the prompts to input the required authentication parameters. These values must match those provided when the Password Vault was created.

NOTE

When prompted for authentication, the keystore password must be given in plaintext form, not masked form.

- Enter *1* (one) to select “Check whether a secured attribute exists”.
- Enter the name of the vault block in which the sensitive string is stored.
- Enter the name of the sensitive string to be checked.

If the sensitive string is stored in the vault block specified, a confirmation message like the following will be output:

```
A value exists for (VAULT_BLOCK, ATTRIBUTE)
```

If the sensitive string is not stored in the specified block, a message like the following will be output:

```
No value has been store for (VAULT_BLOCK, ATTRIBUTE)
```

Example- Check For a Sensitive String Interactively

```
 =========================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 =========================================================================
 **********************************
 ****  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:EAP_HOME/vault
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Oct 22, 2014 12:53:56 PM
org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Vault Configuration in AS7 config file:
 ********************************************
...
```

```
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL"
value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
 ********************************************
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::  0: Store a secured attribute  1: Check whether a
secured attribute exists  2: Remove secured attribute  3: Exit
1
Task: Verify whether a secured attribute exists
Enter Vault Block:vb
Enter Attribute Name:password
A value exists for (vb, password)
Please enter a Digit::  0: Store a secured attribute  1: Check whether a
secured attribute exists  2: Remove secured attribute  3: Exit
```

3.2.1.8. Remove a Sensitive String from the Password Vault

For security reasons it is best to remove sensitive strings from the Password Vault when they are no longer required. For example, if an application is being decommissioned, any sensitive strings used in datasource definitions should be removed at the same time.

IMPORTANT

As a prerequisite, before removing a sensitive string from the Password Vault, confirm if it is used in the configuration of JBoss EAP 6.

This operation can be done either interactively, where the user is prompted for each parameter's value, or non-interactively, where all parameters' values are provided on the commmand line. Each method gives the same result, so either may be used. Both of these methods are invoked using the *vault* script.

Remove a Sensitive String Non-interactively

Use this method to provide all parameters' values at once. For a description of all parameters, see Initialize the Password Vault.

To run the password vault command non-interactively, the *vault* script (located in **`EAP_HOME/bin/`**) can be invoked with parameters for the relavant information:

```
EAP_HOME/bin/vault.sh --keystore KEYSTORE_URL --keystore-password
KEYSTORE_PASSWORD --alias KEYSTORE_ALIAS --remove-sec-attr --vault-block
VAULT_BLOCK --attribute ATTRIBUTE --enc-dir ENC_FILE_DIR --iteration
ITERATION_COUNT --salt SALT
```

Substitute the placeholder values with the actual values. The values for parameters KEYSTORE_URL, KEYSTORE_PASSWORD and KEYSTORE_ALIAS must match those provided when the Password Vault was created.

NOTE

The keystore password must be given in plaintext form, not masked form.

If the sensitive string is successfully removed, a confirmation message like the following will be displayed:

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] has been successfully removed
from vault
```

If the sensitive string is not removed, a message like the following will be displayed:

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] was not removed from vault,
check whether it exist
```

Example Output

```
 ./vault.sh --keystore EAP_HOME/vault/vault.keystore --keystore-password
vault22 --alias vault --remove-sec-attr --vault-block vb --attribute
password --enc-dir EAP_HOME/vault/ --iteration 120 --salt 1234abcd
 =========================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 =========================================================================
Dec 23, 2014 1:54:24 PM org.picketbox.plugins.vault.PicketBoxSecurityVault
init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Secured attribute [vb::password] has been successfully removed from vault
```

Remove a Sensitive String Interactively

To run the password vault command interactively, the following steps are required:

1. Launch the Password Vault command interactively
2. Complete the prompted parameters

1. Launch the Password Vault command interactively

Run **`EAP_HOME/bin/vault.sh`** (on Red Hat Enterprise Linux and similar operating systems) or **`EAP_HOME\bin\vault.bat`** (on Microsoft Windows Server). Start a new interactive session by typing *0* (zero).

2. Complete the Prompted Parameters

Follow the prompts to input the required authentication parameters. These values must match those provided when the Password Vault was created.

NOTE

When prompted for authentication, the keystore password must be given in plaintext form, not masked form.

- Enter *2* (two) to choose option Remove secured attribute.
- Enter the name of the vault block in which the sensitive string is stored.
- Enter the name of the sensitive string to be removed.

If the sensitive string is successfully removed, a confirmation message like the following will be displayed:

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] has been successfully removed
from vault
```

If the sensitive string is not removed, a message like the following will be displayed:

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] was not removed from vault,
check whether it exist
```

Example Output

```
 **********************************
 ****  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:EAP_HOME/vault/
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Dec 23, 2014 1:40:56 PM org.picketbox.plugins.vault.PicketBoxSecurityVault
init
INFO: PBOX000361: Default Security Vault Implementation Initialized and
Ready
Vault Configuration in configuration file:
 *******************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL"
value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
 *******************************************
Vault is initialized and ready for use
Handshake with Vault complete
```

```
Please enter a Digit::  0: Store a secured attribute  1: Check whether a
secured attribute exists  2: Remove secured attribute  3: Exit
2
Task: Remove secured attribute
Enter Vault Block:vb
Enter Attribute Name:password
Secured attribute [vb::password] has been successfully removed from vault
```

3.2.1.9. Configure Red Hat JBoss Enterprise Application Platform 6 to Use a Custom Implementation of the Password Vault

In addition to using the provided Password Vault implementation, a custom implementation of SecurityVault may also be used.

IMPORTANT

As a prerequisite, please ensure that the Password Vault has been initialized. For more information, please see Initialize the Password Vault.

To use a custom implementation for the Password vault:

1. Create a class that implements the interface *SecurityVault*.

2. Create a module containing the class from the previous step, and specify a dependency on *org.picketbox* where the interface is *SecurityVault*.

3. Enable the custom Password Vault in the JBoss EAP 6 configuration by adding the vault element with the following attributes:

    - **code** - The fully qualified name of class that implements SecurityVault.

    - **module** - The name of the module that contains the custom class.

Optionally, the *vault-options* parameters can be used to initialize the custom class for a Password Vault.

Example - Use vault-options Parameters to Initialize the Custom Class

```
/core-service=vault:add( \
code="custom.vault.implementation.CustomSecurityVault", \
module="custom.vault.module", 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")])
```

3.2.1.10. Obtain Keystore Password From External Source

The EXT, EXTC, CMD, CMDC or CLASS methods can be used in Vault configuration for obtaining the Java keystore password.

```
<vault-option name="KEYSTORE_PASSWORD" value="[here]"
```

The description for the methods are listed as:

{EXT}…

Refers to the exact command, where '…' is the exact command. For example: *{EXT}/usr/bin/getmypassword --section 1 --query company*, run the **`/usr/bin/getmypassword`** command, which displays the password on standard output and use it as password for Security Vault's keystore. In this example, the command is using two options: *--section 1* and *--query company*.

{EXTC[:expiration_in_millis]}…

Refers to the exact command, where the *…* is the exact command line that is passed to the *Runtime.exec(String)* method to execute a platform command. The first line of the command output is used as the password. EXTC variant caches the passwords for *expiration_in_millis* milliseconds. Default cache expiration is *0 = infinity*. For example: *{EXTC:120000}/usr/bin/getmypassword --section 1 --query company* verifies if the cache contains **`/usr/bin/getmypassword`** output, if it contains the output then use it. If it does not contain the output, run the command to output it to cache and use it. In this example, the cache expires in 2 minutes (120000 milliseconds).

{CMD}… or {CMDC[:expiration_in_millis]}…

The general command is a string delimited by *,* (comma) where the first part is the actual command and further parts represents the parameters. The comma can be backslashed to keep it as a part of the parameter. For example, *{CMD}/usr/bin/getmypassword,--section,1,--query,company*.

{CLASS[@jboss_module_spec]}classname[:ctorargs]

Where the *[:ctorargs]* is an optional string delimited by the *:* (colon) from the classname is passed to the classname *ctor*. The *ctorargs* is a comma delimited list of strings. For example, *{CLASS@org.test.passwd}org.test.passwd.ExternamPassworProvider*. In this example, the *org.test.passwd.ExternamPassworProvider* class is loaded from *org.test.passwd* module and uses the *toCharArray()* method to get the password. If *toCharArray()* is not available the *toString()* method is used. The *org.test.passwd.ExternamPassworProvider* class must have the default constructor.

3.3. ROLE-BASED ACCESS CONTROL

The basics of or Role-Based Access Control are covered in the Role-Based Access Control and Adding RBAC to the Management Interfaces sections of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide.

3.3.1. Enabling Role-Based Access Control

By default the Role-Based Access Control (RBAC) system is disabled. It is enabled by changing the *provider* attribute from *simple* to *rbac*. *provider* is attribute of the *access-control* element of the *management* element. This can be done using the Management CLI or by editing the server configuration XML file if the server is offline. When RBAC is disabled or enabled on a running server, the server configuration must be reloaded before it takes effect.

Once enabled it can only be disabled by a user of the *Administrator* or *SuperUser* roles. By default the Management CLI runs as the *SuperUser* role if it is run on the same machine as the server.

To enable RBAC with the Management CLI, use the *write-attribute* operation of the access authorization resource to set the *provider* attribute to *rbac*.

CLI to Enable RBAC

```
/core-service=management/access=authorization:write-attribute( \
name=provider, value=rbac)
```

To disable RBAC with the Management CLI, use the *write-attribute* operation of the access authorization resource to set the *provider* attribute to *simple*.

CLI to Disable RBAC

```
/core-service=management/access=authorization:write-attribute( \
name=provider, value=simple)
```

If the server is offline the XML configuration can be edited to enable or disable RBAC. To do this, edit the provider attribute of the access-control element of the management element. Set the value to *rbac* to enable, and *simple* to disable.

Example XML

```
<management>
  <access-control provider="rbac">
    <role-mapping>
      <role name="SuperUser">
        <include>
          <user name="$local"/>
        </include>
      </role>
    </role-mapping>
  </access-control>
</management>
```

3.3.2. Changing the Permission Combination Policy

The Permission Combination Policy determines how permissions are determined if a user is assigned more than one role. This can be set to permissive or rejecting. The default is *permissive*.

When set to *permissive*, if any role is assigned to the user that permits an action, then the action is allowed.

When set to *rejecting*, if multiple roles are assigned to a user, then no action is allowed. This means that when the policy is set to rejecting each user should only be assigned one role. Users with multiple roles will not be able to use the Management Console or the Management CLI when the policy is set to rejecting.

The Permission Combination Policy is configured by setting the *permission-combination-policy* attribute to either *permissive* or *rejecting*. This can be done using the Management CLI or by editing the server configuration XML file if the server is offline. The *permission-combination-policy* attribute is part of the *access-control* element and the *access-control* element can be found in the *management* element.

Setting the Permission Combination Policy

Use the write-attribute operation of the access authorization resource to set the permission-combination-policy attribute to the required policy name.

```
/core-service=management/access=authorization:write-attribute( \
name=permission-combination-policy, value=POLICYNAME)
```

The valid policy names are *rejecting* and *permissive*.

Example CLI

```
/core-service=management/access=authorization:write-attribute( \
name=permission-combination-policy, value=rejecting)
```

If the server is offline the XML configuration can be edited to change the permission combination policy value. To do this, edit the permission-combination-policy attribute of the access-control element.

Example XML

```
<access-control provider="rbac" permission-combination-policy="rejecting">
  <role-mapping>
    <role name="SuperUser">
      <include>
        <user name="$local"/>
      </include>
    </role>
  </role-mapping>
</access-control>
```

3.3.3. Managing Roles

When Role-Based Access Control (RBAC) is enabled, what a management user is permitted to do is determined by the roles to which the user is assigned. JBoss EAP 6 uses a system of includes and excludes based on both the user and group membership to determine to which role a user belongs.

A user is considered to be assigned to a role if the user is:

- listed as a user to be included in the role, or
- a member of a group that is listed to be included in the role.

A user is also considered to be assigned to a role if the user is not:

- listed as a user to exclude from the role, or
- a member of a group that is listed to be excluded from the role.

Exclusions take priority over inclusions.

Role include and exclude settings for users and groups can be configured using both the Management Console and the Management CLI.

Only users of the *SuperUser* or *Administrator* roles can perform this configuration.

3.3.3.1. Configure User Role Assignment using the Management CLI

The configuration of mapping users and groups to roles is located at: */core-service=management/access=authorization* as *role-mapping* elements.

Only users of the *SuperUser* or *Administrator* roles can perform this configuration.

Viewing Role Assignment Configuration

Use the *:read-children-names* operation to get a complete list of the configured roles:

```
/core-service=management/access=authorization:read-children-names(child-
type=role-mapping)
{
    "outcome" => "success",
    "result" => [
        "Administrator",
        "Deployer",
        "Maintainer",
        "Monitor",
        "Operator",
        "SuperUser"
    ]
}
```

Use the *read-resource* operation of a specified role-mapping to get the full details of a specific role:

```
/core-service=management/access=authorization/role-mapping=ROLENAME:read-
resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
```

Add a new role

This procedure shows how to add a role-mapping entry for a role. This must be done before the role can be configured.

Use the *add* operation to add a new role configuration.

```
/core-service=management/access=authorization/role-mapping=ROLENAME:add
```

- ROLENAME is the name of the role that the new mapping is for (e.g. Auditor).

Example CLI

```
/core-service=management/access=authorization/role-mapping=Auditor:add
```

Add a user as included in a role

This procedure shows how to add a user to the included list of a role.

If no configuration for a role has been done, then a role-mapping entry for it must be done first.

Use the *add* operation to add a user entry to the includes list of the role.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include= \
ALIAS:add(name=USERNAME, type=USER)
```

- ROLENAME is the name of the role being configured. (e.g. Auditor)
- ALIAS is a unique name for this mapping. Red Hat recommends that the use of a naming convention for aliases such as user-USERNAME. (e.g. user-max)
- USERNAME is the name of the user being added to the include list. (e.g. max)

Example CLI

```
/core-service=management/access=authorization/role-
mapping=Auditor/include= \
user-max:add(name=max, type=USER)
```

Add a user as excluded in a role

This procedure shows how to add a user to the excluded list of a role.

If no configuration for a role has been done, then a role-mapping entry for it must be done first.

Use the *add* operation to add a user entry to the excludes list of the role.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude= \
ALIAS:add(name=USERNAME, type=USER)
```

- ROLENAME is the name of the role being configured. (e.g. Auditor)
- USERNAME is the name of the user being added to the exclude list. (e.g. max)
- ALIAS is a unique name for this mapping. Red Hat recommends that the use of a naming convention for aliases such as user-USERNAME. (e.g. user-max)

Example CLI

```
/core-service=management/access=authorization/role-
mapping=Auditor/exclude= \
user-max:add(name=max, type=USER)
```

Remove user role include configuration

This procedure shows how to remove a user include entry from a role mapping.

Use the *remove* operation to remove the entry.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include= \
ALIAS:remove
```

- ROLENAME is the name of the role being configured (e.g. Auditor)
- ALIAS is a unique name for this mapping. Red Hat recommends that the use of a naming convention for aliases such as user-USERNAME. (e.g. user-max)

Example CLI

```
/core-service=management/access=authorization/role-
mapping=Auditor/include= \
user-max:remove
```

NOTE

Removing the user from the list of includes does not remove the user from the system, nor does it guarantee that the role won’t be assigned to the user. The role might still be assigned based on group membership.

Remove user role exclude configuration

This procedure shows how to remove an user exclude entry from a role mapping.

Use the remove operation to remove the entry.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude= \
ALIAS:remove
```

- ROLENAME is the name of the role being configured. (e.g. Auditor)
- ALIAS is a unique name for this mapping. Red Hat recommends that the use of a naming convention for aliases such as user-USERNAME. (e.g. user-max)

```
/core-service=management/access=authorization/role-
mapping=Auditor/exclude= \
user-max:remove
```

NOTE

Removing the user from the list of excludes does not remove the user from the system, nor does it guarantee the role will be assigned to the user. Roles might still be excluded based on group membership.

3.3.4. Roles and User Groups

Users authenticated using either the **`mgmt-users.properties`** file or an LDAP server, can be members of user groups. A user group is an arbitrary label that can be assigned to one or more users.

The RBAC system can be configured to automatically assign roles to users depending on what user groups they are members of. It can also exclude users from roles based on group membership.

When using the **`mgmt-users.properties`** file, group information is stored in the **`mgmt-groups.properties`** file. When using LDAP the group information is stored in the LDAP sever and maintained by those responsible for the LDAP server.

3.3.5. Configure Group Role Assignment using the Management CLI

Groups to be included or excluded from a role can be configured in the Management Console and the Management CLI. This topic only shows using the Management CLI.

The configuration of mapping users and groups to roles is located in the management API at: */core-service=management/access=authorization* as *role-mapping* elements.

Only users in the *SuperUser* or *Administrator* roles can perform this configuration.

Viewing Group Role Assignment Configuration

Use the *read-children-names* operation to get a complete list of the configured roles:

```
/core-service=management/access=authorization:read-children-names(child-
type=role-mapping)
{
    "outcome" => "success",
    "result" => [
        "Administrator",
        "Deployer",
        "Maintainer",
        "Monitor",
        "Operator",
        "SuperUser"
    ]
}
```

Use the *read-resource* operation of a specified role-mapping to get the full details of a specific role:

```
/core-service=management/access=authorization/role-mapping= \
ROLENAME:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
```

Add a new role

This procedure shows how to add a role-mapping entry for a role. This must be done before the role can be configured.

Use the *add* operation to add a new role configuration.

```
/core-service=management/access=authorization/role-mapping=ROLENAME:add
```

Add a Group as included in a role

This procedure shows how to add a Group to the included list of a role.

If no configuration for a role has been done, then a role-mapping entry for it must be done first.

Use the add operation to add a Group entry to the includes list of the role.

```
/core-service=management/access=authorization/role-mapping= \
ROLENAME/include=ALIAS:add(name=GROUPNAME, type=GROUP)
```

- ROLENAME is the name of the role being configured. (e.g. Auditor)
- GROUPNAME is the name of the group being added to the include list. (e.g. investigators)
- ALIAS is a unique name for this mapping. Red Hat recommends that you use a naming convention for your aliases such as group-GROUPNAME. (e.g. group-investigators)

Example CLI

```
/core-service=management/access=authorization/role-
mapping=Auditor/include= \
group-investigators:add(name=investigators, type=GROUP)
```

Add a group as excluded in a role

This procedure shows how to add a group to the excluded list of a role.

If no configuration for a role has been done, then a role-mapping entry for it must be created first.

Use the add operation to add a group entry to the excludes list of the role.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude= \
ALIAS:add(name=GROUPNAME, type=GROUP)
```

- ROLENAME is the name of the role being configured (e.g. Auditor)
- GROUPNAME is the name of the group being added to the include list (e.g. supervisors)

- ALIAS is a unique name for this mapping. Red Hat recommends that you use a naming convention for your aliases such as group-GROUPNAME. (e.g. group-supervisors)

Example CLI

```
/core-service=management/access=authorization/role-
mapping=Auditor/exclude= \
group-supervisors:add(name=supervisors, type=GROUP)
```

Remove group role include configuration

This procedure shows how to remove a group include entry from a role mapping.

Use the remove operation to remove the entry.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include= \
ALIAS:remove
```

- ROLENAME is the name of the role being configured (e.g. Auditor)
- ALIAS is a unique name for this mapping. Red Hat recommends that you use a naming convention for your aliases such as group-GROUPNAME. (e.g. group-investigators)

Example CLI

```
/core-service=management/access=authorization/role-
mapping=Auditor/include= \
group-investigators:remove
```

NOTE

Removing the group from the list of includes does not remove the group from the system, nor does it guarantee that the role won't be assigned to users in this group. The role might still be assigned to users in the group individually.

Remove a user group exclude entry

This procedure shows how to remove a group exclude entry from a role mapping.

Use the remove operation to remove the entry.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude= \
ALIAS:remove
```

- ROLENAME is the name of the role being configured. (e.g. Auditor)
- ALIAS is a unique name for this mapping. Red Hat recommends that you use a naming convention for your aliases such as group-GROUPNAME. (e.g. group-supervisors)

```
/core-service=management/access=authorization/role-
mapping=Auditor/exclude= \
group-supervisors:remove
```

NOTE

Removing the group from the list of excludes does not remove the group from the system. It also does not guarantee the role will be assigned to members of the group. Roles might still be excluded based on group membership.

3.3.6. Using RBAC with LDAP

The basics of using RBAC with LDAP as well as how to configure JBoss EAP 6 to use RBAC with LDAP are covered in the *LDAP and RBAC* section in the How to Configure Identity Management guide.

3.3.7. Scoped Roles

Scoped Roles are user-defined roles that grant the permissions of one of the standard roles but only for one or more specified server groups or hosts in an JBoss EAP 6 domain. Scoped roles allow for management users to be granted permissions that are limited to only those server groups or hosts that are required.

IMPORTANT

Scoped roles can be created by users assigned the *Administrator* or *SuperUser* roles.

They are defined by five characteristics:

- A unique name.
- Which of the standard roles it is based on.
- If it applies to Server Groups or Hosts
- The list of server groups or hosts that it is restricted to.
- If all users are automatically included. This defaults to false.

Once created a scoped role can be assigned to users and groups the same way that the standard roles are.

Creating a scoped role does not allow for defining new permissions. Scoped roles can only be used to apply the permissions of an existing role in a limited scope. For example, a scoped role could be created based on the *Deployer* role which is restricted to a single server group.

There are only two scopes that roles can be limited to:

Host-scoped roles

A role that is host-scoped restricts the permissions of that role to one or more hosts. This means access is provided to the relevant */host=*/* resource trees but resources that are specific to other hosts are hidden.

Server-Group-scoped roles

A role that is server-group-scoped restricts the permissions of that role to one or more server groups. Additionally the role permissions will also apply to the profile, socket binding group, server config and server resources that are associated with the specified *server-groups*. Any sub-resources within any of those that are not logically related to the server-group will not be visible to the user.

3.3.7.1. Configuring Scoped Roles from the Management CLI

IMPORTANT

Only users in the *SuperUser* or *Administrator* roles can perform this configuration.

Add a New Scoped Role

To add a new Scoped Role, the following operations must be done:

```
/core-service=management/access=authorization/server-group-scoped-role= \
NEW-SCOPED-ROLE:add(base-role=BASE-ROLE, server-groups=[SERVER-GROUP-
NAME])
```

```
/core-service=management/access=authorization/role-mapping=NEW-SCOPED-
ROLE:add
```

Replace NEW-SCOPED-ROLE, BASE-ROLE, and SERVER-GROUP-NAME with the proper information.

Viewing and Editing a Scoped Role Mapping

A Scoped Role's details (including members) can be view by issuing the following command:

```
/core-service=management/access=authorization/role-mapping= \
NEW-SCOPED-ROLE:read-resource(recursive=true)
```

Replace NEW-SCOPED-ROLE with the proper information.

To edit a Scoped Role's details, the *write-attribute* command may be used. For example:

```
/core-service=management/access=authorization/role-mapping= \
NEW-SCOPED-ROLE:write-attribute(name=include-all, value=true)
```

Replace NEW-SCOPED-ROLE with the proper information.

Delete a Scoped Role

```
/core-service=management/access=authorization/role-mapping= \
NEW-SCOPED-ROLE:remove
```

```
/core-service=management/access=authorization/server-group-scoped-role= \
NEW-SCOPED-ROLE:remove
```

Replace NEW-SCOPED-ROLE with the proper information.

IMPORTANT

A Scoped Role cannot be deleted if users or groups are assigned to it. Remove the role assignments first, and then delete it.

Adding and Removing Users

Adding and removing users to and from Scoped Roles follows the same process as adding and removing standard roles.

3.3.7.2. Configuring Scoped Roles from the Management Console

IMPORTANT

Only users in the *SuperUser* or *Administrator* roles can perform this configuration.

Scoped Role configuration in the management console can be found by following these steps:

1. Login to the Management Console
2. Click on the *Administration* tab
3. Under the *Access Control* menu on the left and select *Role Assignment*.
4. Select *ROLES* tab, and then the *Scoped Roles* tab within it.

The Scoped Roles section of the Management Console consists of two main areas, a table containing a list of the currently configured scoped roles, and the *Selection* panel which displays the details of the role currently selected in the table.

The following procedures show how to perform configuration tasks for Scoped Roles.

Add a New Scoped Role

1. Login to the Management Console
2. Navigate to the *Scoped Roles* area of the *Roles* tab.
3. Click *Add*. The *Add Scoped Role* dialog appears.
4. Specify the following details:
   - **Name**, the unique name for the new scoped role.
   - **Base Role**, the role which this role will base its permissions on.
   - **Type**, whether this role will be restricted to hosts or server groups.
   - **Scope**, the list of hosts or server groups that the role is restricted to. Multiple entries can be selected.
   - **Include All**, should this role automatically include all users. Defaults to no.
5. Click *Save* and the dialog will close and the newly created role will appear in the table.

Edit a Scoped Role

1. Login to the Management Console
2. Navigate to the *Scoped Roles* area of the *Roles* tab.
3. Click on the desired scoped role to edit in the table. The details of that role appears in the *Selection* panel below the table.
4. Click *Edit* in the *Selection* panel. The *Selection* panel enters edit mode.

5. Update the desired details to change and click the *Save* button. The *Selection* panel returns to its previous state. Both the *Selection* panel and table show the newly updated details.

View Scoped Role Members

1. Login to the Management Console
2. Navigate to the *Scoped Roles* area of the *Roles* tab.
3. Click on the desired scoped role in the table to view the *Members* of, then click *Members*. The *Members* of role dialog appears. It shows users and groups that are included or excluded from the role.
4. Click *Done* when finished reviewing this information.

Delete a Scoped Role

1. Login to the Management Console
2. Navigate to the *Scoped Roles* area of the *Roles* tab.
3. Select the scoped role to be removed in the table.
4. Click the *Remove* button. The *Remove Scoped Role* dialog appears.
5. Click *Confirm*.The dialog closes and the role is removed.

IMPORTANT

A Scoped Role cannot be deleted if users or groups are assigned to it. Remove the role assignments first, and then delete it.

Adding and Removing Users

Adding and removing users to and from Scoped Roles follows the same process as adding and removing standard roles. To update a user’s Scoped Roles:

1. Login to the Management Console
2. Click on the *Administration* tab
3. Under the *Access Control* menu on the left and select *Role Assignment*.
4. Click on the *Users* tab.
5. Select the desired user from the table.
6. Click *Edit* in the *Selection* section
7. Once the desired roles have been added/removed, click *Save*.

3.3.8. Configuring Constraints

3.3.8.1. Configuring Sensitivity Constraints

Each Sensitivity Constraint defines a set of resources that are considered *sensitive*. A *sensitive* resource is generally one that either should be secret, like passwords, or one that will have serious impact on the server, like networking, JVM configuration, or system properties. The access control system itself is also considered sensitive. Resource sensitivity limits which roles are able to read, write or address a specific resource.

Sensitivity constraint configuration is in the at */core-service=management/access=authorization/constraint=sensitivity-classification*.

Within the management model each Sensitivity Constraint is identified as a classification. The classifications are then grouped into types. There are 39 included classifications that are arranged into 13 types.

To configure a sensitivity constraint, use the *write-attribute* operation to set the *configured-requires-read*, *configured-requires-write*, or *configured-requires-addressable* attribute. To make that type of operation sensitive set the value of the attribute to *true*, otherwise to make it nonsensitive set it to *false*. By default these attributes are not set and the values of *default-requires-read*, *default-requires-write*, and *default-requires-addressable* are used. Once the configured attribute is set it is that value that is used instead of the default. The default values cannot be changed.

Example - Make reading system properties a sensitive operation

```
/core-service=management/access=authorization/constraint= \
sensitivity-classification/type=core/classification= \
system-property:write-attribute(name=configured-requires-read, \
value=true)
```

Result

```
/core-service=management/access=authorization/constraint= \
sensitivity-classification/type=core/classification= \
system-property:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "configured-requires-addressable" => undefined,
        "configured-requires-read" => true,
        "configured-requires-write" => undefined,
        "default-requires-addressable" => false,
        "default-requires-read" => false,
        "default-requires-write" => true,
        "applies-to" => {
            "/core-service=platform-mbean/type=runtime" => undefined,
            "/system-property=*" => undefined,
            "/" => undefined
        }
    }
}
```

What roles will be able to perform what operations depending on the configuration of these attributes is summarized in the following table:

Table 3.1. Sensitivity Constraint Configuration Outcomes

Value	requires-read	requires-write	requires-addressable
true	Read is sensitive. Only Auditor, Administrator, SuperUser can read.	Write is sensitive. Only Administrator and SuperUser can write	Addressing is sensitive. Only Auditor, Administrator, SuperUser can address.
false	Read is not sensitive. Any management user can read.	Write is not sensitive. Only Maintainer, Administrator and SuperUser can write. Deployers can also write the resource is an application resource.	Addressing is not sensitive. Any management user can address.

3.3.8.2. Configure Application Resource Constraints

Each Application Resource Constraint defines a set of resources, attributes and operations that are usually associated with the deployment of applications and services. When an application resource constraint is enabled management users of the *Deployer* role are granted access to the resources that it applies to.

Application constraint configuration is at */core-service=management/access=authorization/constraint=application-classification/*.

Each Application Resource Constraint is identified as a classification. The classifications are then grouped into types. There are 14 included classifications that are arranged into 8 types. Each classification has an *applies-to* element which is a list of resource path patterns to which the classifications configuration applies.

By default the only Application Resource classification that is enabled is core. Core includes deployments, deployment overlays, and the deployment operations.

To enable an Application Resource, use the *write-attribute* operation to set the *configured-application* attribute of the classification to *true*. To disable an Application Resource, set this attribute to *false*. By default these attributes are not set and the value of *default-application* attribute is used. The default value cannot be changed.

Enabling the logger-profile application resource classification

```
/core-service=management/access=authorization/constraint= \
application-classification/type=logging/classification= \
logging-profile:write-attribute(name=configured-application, \
value=true)
```

Result

```
/core-service=management/access=authorization/constraint= \
application-classification/type=logging/classification= \
logging-profile:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "configured-application" => true,
        "default-application" => false,
        "applies-to" => {"/subsystem=logging/logging-profile=*" =>
undefined}
    }
}
```

IMPORTANT

Application Resource Constraints apply to all resources that match its configuration. For example, it is not possible to grant a *Deployer* user access to one datasource resource but not another. If this level of separation is required then it is recommended to configure the resources in different server groups and create different scoped *Deployer* roles for each group.

3.3.8.3. Configure the Vault Expression Constraint

By default, reading and writing vault expressions are sensitive operations. Configuring the Vault Expression Constraint allows either or both of those operations to be set to nonsensitive. Changing this constraint allows a greater number of roles to read and write vault expressions.

The vault expression constraint is found in the at */core-service=management/access=authorization/constraint=vault-expression*.

To configure the vault expression constraint, use the *write-attribute* operation to set the attributes of *configured-requires-write* and *configured-requires-read* to *true* or *false*. By default these are not set and the values of *default-requires-read* and *default-requires-write* are used. The default values cannot be changed.

Making writing to vault expressions a nonsensitive operation

```
/core-service=management/access=authorization/constraint= \
vault-expression:write-attribute(name=configured-requires-write, \
value=false)
```

Result

```
/core-service=management/access=authorization/constraint= \
vault-expression:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "configured-requires-read" => undefined,
        "configured-requires-write" => false,
        "default-requires-read" => true,
        "default-requires-write" => true
    }
}
```

What roles will be able to read and write to vault expressions depending on this configuration is summarized in the following table:

Table 3.2. Vault Expression Constraint Configuration outcomes

Value	requires-read	requires-write
true	Read operation is sensitive. Only Auditor, Administrator, and SuperUser can read.	Write operation is sensitive. Only Administrator, and SuperUser can write.
false	Read operation is not sensitive. All management users can read.	Write operation is not sensitive. Monitor, Administrator, and SuperUser can write. Deployers can also write if the vault expression is in an Application Resource.

3.3.8.4. Application Resource Constraints Reference

Type: core - Classification: deployment-overlay

- default: true
- PATH: /deployment-overlay=*
- PATH: /deployment=*
- PATH: /
- Operation: upload-deployment-stream, full-replace-deployment, upload-deployment-url, upload-deployment-bytes

Type: datasources - Classification: datasource

- default: false
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=ExampleDS
- PATH: /deployment=*/subsystem=datasources/data-source=*

Type: datasources - Classification: jdbc-driver

- default: false
- PATH: /subsystem=datasources/jdbc-driver=*

Type: datasources - Classification: xa-data-source

- default: false
- PATH: /subsystem=datasources/xa-data-source=*

- PATH: /deployment=*/subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/xa-data-source=*

Type: logging - Classification: logger

- default: false
- PATH: /subsystem=logging/logger=*
- PATH: /subsystem=logging/logging-profile=*/logger=*

Type: datasources - Classification: logging-profile

- default: false
- PATH: /subsystem=logging/logging-profile=*

Type: mail - Classification: mail-session

- default: false
- PATH: /subsystem=mail/mail-session=*

Type: naming - Classification: binding

- default: false
- PATH: /subsystem=naming/binding=*

Type: resource-adapters - Classification: resource-adapters

- default: false
- PATH: /subsystem=resource-adapters/resource-adapter=*

Type: security - Classification: security-domain

- default: false
- PATH: /subsystem=security/security-domain=*

3.3.8.5. Sensitivity Constraints Reference

Type: core - Classification: access-control

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/access=authorization
- PATH: /subsystem=jmx ATTRIBUTE: non-core-mbean-sensitivity

Type: core - Classification: credential

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username , password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username , password
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, recovery-username, password, recovery-password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, password
- PATH: /subsystem=datasources/data-source=*" ATTRIBUTE: user-name, password
- PATH: /subsystem=remoting/remote-outbound-connection=*" ATTRIBUTE: username
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, password
- PATH: /subsystem=web/connector=*/configuration=ssl ATTRIBUTE: key-alias, password
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*" ATTRIBUTE: recovery-username, recovery-password

Type: core - Classification: domain-controller

- requires-addressable: false
- requires-read: false
- requires-write: true

Type: core - Classification: domain-names

- requires-addressable: false
- requires-read: false
- requires-write: true

Type: core - Classification: extensions

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /extension=*

Type: core - Classification: jvm

- requires-addressable: false

- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: input-arguments, boot-class-path, class-path, boot-class-path-supported, library-path

Type: core - Classification: management-interfaces

- requires-addressable: false
- requires-read: false
- requires-write: true
- /core-service=management/management-interface=native-interface
- /core-service=management/management-interface=http-interface

Type: core - Classification: module-loading

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=module-loading

Type: core - Classification: patching

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=patching/addon=*
- PATH: /core-service=patching/layer=*"
- PATH: /core-service=patching

Type: core - Classification: read-whole-config

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: / OPERATION: read-config-as-xml

Type: core - Classification: security-domain

- requires-addressable: true

- requires-read: true
- requires-write: true
- PATH: /subsystem=security/security-domain=*

Type: core - Classification: security-domain-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=ejb3 ATTRIBUTE: default-security-domain
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=* ATTRIBUTE: security-domain, recovery- security-domain, security-application, security-domain-and-application

Type: core - Classification: security-realm

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/security-realm=*

Type: core - Classification: security-realm-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: security-realm

Type: core - Classification: security-vault

- requires-addressable: false
- requires-read: false

- requires-write: true
- PATH: /core-service=vault

Type: core - Classification: service-container

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=service-container

Type: core - Classification: snapshots

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: / ATTRIBUTE: take-snapshot, list-snapshots, delete-snapshot

Type: core - Classification: socket-binding-ref

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=web/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=remoting/local-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /socket-binding-group=*/local-destination-outbound-socket-binding=* ATTRIBUTE: socket-binding-ref
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-binding, status-socket-binding, socket-binding

Type: core - Classification: socket-config

- requires-addressable: false

- requires-read: false
- requires-write: true
- PATH: /interface=* OPERATION: resolve-internet-address
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: port, interface, socket-binding
- PATH: /socket-binding-group=*
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: port, secure-port, interface, secure-socket-binding, socket-binding
- PATH: / OPERATION: resolve-internet-address
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-max-ports

Type: core - Classification: system-property

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: system-properties
- PATH: /system-property=*
- PATH: / OPERATION: resolve-expression

Type: datasources - Classification: data-source-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, security-domain, password
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: user-name, security-domain, password

Type: jdr - Classification: jdr

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jdr OPERATION: generate-jdr-report

Type: jmx - Classification: jmx

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jmx

Type: mail - Classification: mail-server-security

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=/**custom=** ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, tls, ssl, password

Type: naming - Classification: jndi-view

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=naming OPERATION: jndi-view

Type: naming - Classification: naming-binding

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=naming/binding=*

Type: remoting - Classification: remoting-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: authentication-provider, security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: username, security-realm

- PATH: /subsystem=remoting/connector=*/security=sasl

Type: resource-adapters - Classification: resource-adapter-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=* ATTRIBUTE: security-domain, recovery-username, recovery-security-domain, security-application, security-domain-and-application, recovery-password

Type: security - Classification: misc-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=security ATTRIBUTE: deep-copy-subject-mode

Type: web - Classification: web-access-log

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/virtual-server=*/configuration=access-log

Type: web - Classification: web-connector

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/connector=*

Type: web - Classification: web-ssl

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/connector=*/configuration=ssl

Type: web - Classification: web-sso

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/virtual-server=*/configuration=sso

Type: web - Classification: web-valve

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/valve=*