Red Hat JBoss Data Virtualization 6.2

Security Guide

This guide is intended for administrators

Last Updated: 2017-10-20

Red Hat JBoss Data Virtualization 6.2 Security Guide

This guide is intended for administrators

Red Hat Customer Content Services

Abstract

This document provides information on configuring security for JBoss Data Virtualization.

Table of Contents

CHAPTER 1. READ ME

1.1. BACK UP YOUR DATA

WARNING

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

1.2. VARIABLE NAME: EAP_HOME

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

1.3. VARIABLE NAME: MODE

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

1.4. RED HAT DOCUMENTATION SITE

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

CHAPTER 2. JBOSS EAP SECURITY DOMAINS

2.1. ABOUT SECURITY DOMAINS

Security domains are part of the JBoss EAP 6 security subsystem. All security configuration is now managed centrally, by the domain controller of a managed domain, or by the standalone server.

A security domain consists of configurations for authentication, authorization, security mapping, and auditing. It implements *Java Authentication and Authorization Service (JAAS)* declarative security.

Authentication refers to verifying the identity of a user. In security terminology, this user is referred to as a *principal*. Although authentication and authorization are different, many of the included authentication modules also handle authorization.

Authorization is a process by which the server determines if an authenticated user has permission or privileges to access specific resources in the system or operation.

Security mapping refers to the ability to add, modify, or delete information from a principal, role, or attribute before passing the information to your application.

The auditing manager allows you to configure *provider modules* to control the way that security events are reported.

If you use security domains, you can remove all specific security configuration from your application itself. This allows you to change security parameters centrally. One common scenario that benefits from this type of configuration structure is the process of moving applications between testing and production environments.

2.2. ABOUT AUTHENTICATION

Authentication refers to identifying a subject and verifying the authenticity of the identification. The most common authentication mechanism is a username and password combination. Other common authentication mechanisms use shared keys, smart cards, or fingerprints. The outcome of a successful authentication is referred to as a principal, in terms of Java Enterprise Edition declarative security.

JBoss EAP 6 uses a pluggable system of authentication modules to provide flexibility and integration with the authentication systems you already use in your organization. Each security domain may contain one or more configured authentication modules. Each module includes additional configuration parameters to customize its behavior. The easiest way to configure the authentication subsystem is within the web-based management console.

Authentication is not the same as authorization, although they are often linked. Many of the included authentication modules can also handle authorization.

2.3. CONFIGURE AUTHENTICATION IN A SECURITY DOMAIN

To configure authentication settings for a security domain, log into the management console and follow this procedure.

Procedure 2.1. Setup Authentication Settings for a Security Domain

1. **Open the security domain's detailed view.**

    a. Click the `Configuration` label at the top of the management console.

b. Select the profile to modify from the **`Profile`** selection box at the top left of the Profile view.

c. Expand the **`Security`** menu, and select **`Security Domains`**.

d. Click the **`View`** link for the security domain you want to edit.

2. **Navigate to the Authentication subsystem configuration.**
   Select the **`Authentication`** label at the top of the view if it is not already selected.

   The configuration area is divided into two areas: **`Login Modules`** and **`Details`**. The login module is the basic unit of configuration. A security domain can include several login modules, each of which can include several attributes and options.

3. **Add an authentication module.**
   Click **`Add`** to add a JAAS authentication module. Fill in the details for your module.

   The **`Code`** is the class name of the module. The **`Flag`** setting controls how the module relates to other authentication modules within the same security domain.

   Explanation of the Flags

   The Java Enterprise Edition 6 specification provides the following explanation of the flags for security modules. The following list is taken from http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Appendix Refer to that document for more detailed information.

Flag	Details
required	The LoginModule is required to succeed. If it succeeds or fails, authentication still continues to proceed down the LoginModule list.
requisite	LoginModule is required to succeed. If it succeeds, authentication continues down the LoginModule list. If it fails, control immediately returns to the application (authentication does not proceed down the LoginModule list).
sufficient	The LoginModule is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the LoginModule list). If it fails, authentication continues down the LoginModule list.
optional	The LoginModule is not required to succeed. If it succeeds or fails, authentication still continues to proceed down the LoginModule list.

4. **Edit authentication settings**
   After you have added your module, you can modify its **`Code`** or **`Flags`** by clicking **`Edit`** in the **`Details`** section of the screen. Be sure the **`Attributes`** tab is selected.

5. **Optional: Add or remove module options.**

If you need to add options to your module, click its entry in the **`Login Modules`** list, and select the **`Module Options`** tab in the **`Details`** section of the page. Click the **`Add`** button, and provide the key and value for the option. Use the **`Remove`** button to remove an option.

6. Reload the Red Hat JBoss Data Virtualization server to make the authentication module changes available to applications.

Result

Your authentication module is added to the security domain, and is immediately available to applications which use the security domain.

The `jboss.security.security_domain` Module Option

By default, each login module defined in a security domain has the **`jboss.security.security_domain`** module option added to it automatically. This option causes problems with login modules which check to make sure that only known options are defined. The IBM Kerberos login module, **`com.ibm.security.auth.module.Krb5LoginModule`** is one of these.

You can disable the behavior of adding this module option by setting the system property to **`true`** when starting JBoss EAP 6. Add the following to your start-up parameters.

```
-Djboss.security.disable.secdomain.option=true
```

You can also set this property using the web-based Management Console. In a standalone server, you can set system properties in the **`Profile`** section of the configuration. In a managed domain, you can set system properties for each server group.

2.4. ABOUT AUTHORIZATION

Authorization is a mechanism for granting or denying access to a resource based on identity. It is implemented as a set of declarative security roles which can be added to principals.

JBoss EAP 6 uses a modular system to configure authorization. Each security domain may contain one or more authorization policies. Each policy has a basic module which defines its behavior. It is configured through specific flags and attributes. The easiest way to configure the authorization subsystem is by using the web-based management console.

Authorization is different from authentication, and usually happens after authentication. Many of the authentication modules also handle authorization.

2.5. CONFIGURE AUTHORIZATION IN A SECURITY DOMAIN

To configure authorization settings for a security domain, log into the management console and follow this procedure.

Procedure 2.2. Setup Authorization in a Security Domain

1. **Open the security domain's detailed view.**

    a. Click the **`Configuration`** label at the top of the management console.

    b. In a managed domain, select the profile to modify from the **`Profile`** drop down box at the top left.

   c. Expand the **`Security`** menu item, and select **`Security Domains`**.

   d. Click the **`View`** link for the security domain you want to edit.

2. **Navigate to the Authorization subsystem configuration.**
   Select the **`Authorization`** label at the top of the screen.

   The configuration area is divided into two areas: **`Policies`** and **`Details`**. The policy module is the basic unit of configuration. A security domain can include several authorization policies, each of which can include several attributes and options.

3. **Add a policy.**
   Click **`Add`** to add a JAAS authorization policy module. Fill in the details for your module.

   The **`Code`** is the class name of the module. The **`Flag`** controls how the module relates to other authorization policy modules within the same security domain.

   Explanation of the Flags

   The Java Enterprise Edition 6 specification provides the following explanation of the flags for security modules. The following list is taken from http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Appendix Refer to that document for more detailed information.

Flag	Details
required	The policy module is required to succeed. If it succeeds or fails, authorization still continues to proceed down the policy module list.
requisite	The policy module is required to succeed. If it succeeds, authorization continues down the policy module list. If it fails, control immediately returns to the application (authorization does not proceed down the poicy module list).
sufficient	The policy module is not required to succeed. If it does succeed, control immediately returns to the application (authorization does not proceed down the policy module list). If it fails, authorization continues down the policy module list.
optional	The policy module is not required to succeed. If it succeeds or fails, authorization still continues to proceed down the policy module list.

4. **Edit authorization settings**
   After you have added your module, you can modify its **`Code`** or **`Flags`** by clicking **`Edit`** in the **`Details`** section of the screen. Be sure the **`Attributes`** tab is selected.

5. **Optional: Add or remove module options.**
   If you need to add options to your module, click its entry in the **`Policies`** 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. Use the **`Remove`** button to remove an option.

Result

Your authorization policy module is added to the security domain. Reload the server for it to become available.

2.6. ENCRYPTION

Teiid Transports

Teiid provides built-in support for JDBC/ODBC over SSL. JDBC defaults to just sensitive message encryption (login mode), while ODBC (the pg transport) defaults to just clear text passwords if using simple username/password authentication.

The AS instance must be configured for SSL as well so that any web services consuming Teiid may use SSL.

Configuration

Passwords in configuration files are by default stored in plain text. If you need these values to be encrypted, please see encrypting passwords for instructions on encryption facilities provided by the container.

Source Access

Encrypting remote source access is the responsibility for the resource adapter and library/driver used to access the source system.

Temporary Data

Teiid temporary data which can be stored on the file system as configured by the BufferManager may optionally be encrypted. Set the buffer-service-encrypt-files property to true on the Teiid subsystem to use 128-bit AES to encrypt any files written by the BufferManager. A new symmetric key will be generated for each start of the Teiid system on each server. A performance hit will be seen for processing that is memory intensive such that data typically spills to disk. This setting does not affect how VDBs (either the artifact or an exploded form) or log files are written to disk.

2.7. TEMPORARY DATA

You can encrypt the temporary data stored on the file system as configured by the BufferManager.

Set the buffer-service-encrypt-files property to true on the Teiid subsystem to use 128-bit AES to encrypt any files written by the BufferManager. A new symmetric key will be generated for each start of the Teiid system on each server. A performance hit will be seen for processing that is memory intensive such that data typically spills to disk. This setting does not affect how VDBs (either the artifact or an exploded form) or log files are written to disk.

CHAPTER 3. CONFIGURING AUTHENTICATION AND AUTHORIZATION FOR JBOSS DATA VIRTUALIZATION

3.1. CLIENT AUTHENTICATION

Client authentication concerns authenticating users of a JBoss Data Virtualization instance.

JDBC/ODBC/Web Service clients may use simple passwords to authenticate a user.

Typically a user name is required, however user names may be considered optional if the identity of the user can be discerned by the password credential alone. In any case it is up to the configured security domain to determine whether a user can be authenticated. If you need authentication, the administrator must configure an authentication module.

Auto-generated web services, such as OData, typically support HTTPBasic authentication. This should use Pass-through Authentication.

NOTE

Kerberos authentication is also supported.

3.2. DATA SOURCE AUTHENTICATION

Data source authentication is generally determined by the capabilities of JCA resource adapters used to connect to external resources. Consult the JBoss EAP JCA documentation for the capabilities of data source pooling and supplied resource adapters for more information. Typically a single username/password credential is supported, such as when creating JDBC Data Sources. In more advanced usage scenarios the source and/or translator may be configured or customized to use an execution payload, the JBoss Data Virtualization subject, or even the calling application subject via Pass-through Authentication. See more about Developing JEE Connectors and Translator Development in *Red Hat JBoss Data Virtualization Development Guide: Server Development*.

3.3. PASS-THROUGH AUTHENTICATION

If your client application (web application or web service) is deployed on the same application server instance as JBoss Data Virtualization and your client application uses a security domain to handle authentication, you can configure JBoss Data Virtualization to use that same security domain. This way, the user will not have to re-authenticate in order to use JBoss Data Virtualization.

In pass-through mode, Red Hat JBoss Data Virtualization looks for an authenticated subject in the calling thread context and uses it for sessioning and authorization.

Procedure 3.1. Configure Pass-Through Authentication

- Change the Teiid security-domain name in the embedded "transport" section to the same name as your application's security domain name. (You can make this change via the CLI or by editing the standalone.xml file if you are running the product in standalone mode.).

IMPORTANT

The security domain must be a JAAS-based LoginModule and your client application must obtain its Teiid connection using a Local Connection with the PassthroughAuthentication connection flag set to true.

3.4. DATA ROLES

All authenticated users have access to a VDB. To restrict access, configure data roles. Set these in the Teiid Designer or the dynamic VDB's META-INF/vdb.xml file.

As part of the data role definition, you can map them to JAAS roles specified in **`<mapped-role-name>`** tags. (Establish these mappings using the **`addDataRoleMapping()`** method.)

How these JAAS roles are associated with users depends on which particular JAAS login module you use. For instance, the default **`UsersRolesLoginModule`** associates users with JAAS roles in plain text files.

For more information about data roles, see *Red Hat JBoss Data Virtualization Development Guide: Reference Material*.

IMPORTANT

Do not use "admin" or "user" as JAAS role names as these are reserved specifically for Dashboard Builder permissions.

CHAPTER 4. AUTHENTICATION MODULES

4.1. CONFIGURING TRANSPORTS

The **`security-domain`** attribute within the **`transport`** element is used to set a comma separated list of desired security domains (and their associated authentication modules).

```
<transport name="jdbc" protocol="teiid" socket-binding="teiid-jdbc">
     <authentication security-domain="teiid-security"/>
  </transport>
```

Usernames can be fully qualified to authenticate only against a particular domain:

```
username@domainname
```

If a username is not fully qualified, the installed domains will be consulted in order until a domain successfully authenticates the user.

If no domain can authenticate the user, the login attempt will fail. Details of the failed attempt (including information such as invalid users and which domains were consulted) will be in the server log with appropriate levels of severity.

The security domain defined for each transport type can be different. Users can configure a unique transport for JDBC and ODBC (and for multiple JDBC ports), each with a different security domain.

WARNING

In existing installations a security domain may already be configured for use by administrative clients (such as the management concole). If the admin connections (CLI and AdminShell) are not secured, it is recommended that you secure these interfaces by executing the ***`EAP_HOME`*`/bin/scripts/add-user.sh`** script.

4.2. DEFAULT FILE BASED AUTHENTICATION MODULE

The default login module for teiid-security is RealmDirect:

```
 <subsystem xmlns="urn:jboss:domain:security:1.2">
 <security-domain name="teiid-security" cache-type="default">
    <authentication>
        <login-module code="RealmDirect" flag="sufficient">
            <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
    </authentication>
</security-domain>
```

NOTE

This module uses md5 to encrypt passwords.

By default, Red Hat JBoss Data Virtualization inherits its definition of users and roles from EAP. You will find the configuration in the following section of the relevant configuration file (either (standalone/configuration/standalone.xml or domain/configuration/host.xml):

```
<security-realm name="ApplicationRealm">
```

See Also:

- Section 2.3, “Configure Authentication in a Security Domain”

4.3. LDAP BASED AUTHENTICATION MODULE

The following is an example of what would appear in the server configuration file for an **LDAPExtended** authentication module defined for the **ldap_security_domain** security domain.

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="ldap_security_domain">
            <authentication>
                <login-module code="LdapExtended" flag="required">
                    <module-option name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
                    <module-option name="java.naming.provider.url"
value="ldap://mydomain.org:389" />
                    <module-option
name="java.naming.security.authentication" value="simple" />
                    <module-option name="bindDN" value="myuser" />
                    <module-option name="bindCredential" value="mypasswd"
/>
                    <module-option name="baseCtxDN"
value="ou=People,dc=XXXX,dc=ca" />
                    <module-option name="baseFilter" value="(cn={0})" />
                    <module-option name="rolesCtxDN" value="ou=Webapp-
Roles,ou=Groups,dc=XXXX,dc=ca" />
                    <module-option name="roleFilter" value="(member={1})"
/>
                    <module-option name="uidAttributeID" value="member"
/>
                    <module-option name="roleAttributeID" value="cn" />
                    <module-option name="roleAttributeIsDN" value="true"
/>
                    <module-option name="roleNameAttributeID" value="cn"
/>
                    <module-option name="roleRecursion" value="-1" />
                    <module-option name="searchScope"
value="ONELEVEL_SCOPE" />
                    <module-option name="allowEmptyPasswords"
value="false" />
                    <module-option name="throwValidateError" value="true"
/>
```

```
                </login-module>
            </authentication>
        </security-domain>
    </security-domains>
</subsystem>
```

For more information about EAP security, the **`LDAPExtended`** module, or any of the provided authentication modules, refer to the JBoss Enterprise Application Platform *Security Guide*.

NOTE

If using SSL to connect to the LDAP server, you would also need to ensure the Corporate CA Certificate is added to the JRE truststore.

See Also:

- Section 2.3, “Configure Authentication in a Security Domain”

4.4. ROLE MAPPING LOGINMODULE

If the LoginModule you are using exposes role names that you wish to map to more application specific names, then you can use the RoleMappingLoginModule. This uses a properties file to inject additional role names, and optionally replace the existing role, on authenticated subjects. This is what the security domain should look like:

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="jdv_security_domain">
            <authentication>
                ...
                <login-module
code="org.jboss.security.auth.spi.RoleMappingLoginModule" flag="optional">
                    <module-option name="rolesProperties"
value="${jboss.server.base.dir}/configuration/roles.properties" />
                    <module-option name="replaceRole" value="false" />
                </login-module>
                ...
            </authentication>
        </security-domain>
    </security-domains>
</subsystem>
```

4.5. CUSTOM AUTHENTICATION MODULES

A custom authentication module may be a subclass of a provided module or a completely new module.

All authentication modules implement the javax.security.auth.spi.LoginModule interface. Refer to the relevant API documentation for more information.

4.6. EXAMPLE CUSTOM AUTHENTICATION MODULE

Suppose you are working on a project where user names and passwords are stored in a relational database; however, the passwords are base64 encoded, so you can't use the **`DatabaseServerLoginModule`** module directly. You can provide a subclass:

```
public class MyLoginModule
    extends DatabaseServerLoginModule
{
   protected String convertRawPassword(String password)
   {
        try {
            return new String((new
sun.misc.BASE64Decoder()).decodeBuffer(password));
        } catch (IOException e) {
            return password;
        }
   }
}
```

To use this new module, you will need to declare a new security domain in the server configuration file:

```
<security-domain name="my-security-domain">
    <authentication>
        <login-module code="com.mycompany.MyLoginModule" flag="required">
            <module-option name="dsJndiName">java:MyDataSource</module-
option>
            <module-option name="principalsQuery">select password from
usertable where login=?</module-option>
            <module-option name="rolesQuery">select role, 'Roles' from
users, userroles where login=? and users.roleId=userroles.roleId</module-
option>
        </login-module>
    </authentication>
</security-domain>
```

After that, configure the transport to use the security domain with the new authentication module:

```
<transport name="jdbc" protocol="teiid" socket-binding="teiid-jdbc">
 <authentication security-domain="my-security-domain"/>
</transport>
```

NOTE

DatabaseServerLoginModule is in the picketbox JAR (moved from jbosssx in earlier versions).

Maven pom.xml dependency example for Red Hat JBoss EAP:

```
<dependency>
    <groupId>org.picketbox</groupId>
    <artifactId>picketbox</artifactId>
    <version>4.1.1.Final-redhat-1</version>
    <scope>provided</scope>
 </dependency>
```

4.7. CONFIGURE PASS-THROUGH AUTHENTICATION

Procedure 4.1. Configure Pass-Through Authentication

- **Set the security domain**
  Change the JBoss Data Virtualization security domain to the same name as your application's security domain name in the `transport` section of the server configuration file.

NOTE

For this to work, the security domain must be a JAAS based login module and your client application must obtain its JBoss Data Virtualization connection using a local connection with the `PassthroughAuthentication=true` connection flag set.

CHAPTER 5. AUTHORIZATION MODULES

5.1. CUSTOM AUTHORIZATION MODULES

In situations where the JBoss Data Virtualization built-in role mechanism is not sufficient, a custom **`org.teiid.PolicyDecider`** can be installed via a JBoss module. This is a two-stage process: first you must create a jar that contains your custom class, then you must expose the jar as a JBoss module so it can be seen by all of your classes.

1. Implement the org.teiid.PolicyDecider interface and build a custom java class. If you are using maven as your build process, you can use the following dependencies:

```
<dependencies>
      <dependency>
          <groupId>org.jboss.teiid</groupId>
             <artifactId>teiid-api</artifactId>
        <scope>provided</scope>
      </dependency>
      <dependency>
          <groupId>org.jboss.teiid</groupId>
             <artifactId>teiid-common-core</artifactId>
        <scope>provided</scope>
      </dependency>
</dependencies>
```

2. The PolicyDecider interface is loaded by JBoss Data Virtualization using Java's standard service loader mechanism. For this to work, add the **`META-INF/services/org.teiid.PolicyDecider`** file with the full name of your PolicyDecider implementation class as its contents. Here is an example:

```
org.jboss.teiid.auth.MyCustomPolicyDecider
```

3. Package all of these files into a JAR archive.

4. Create a directory called **`[JDV_HOME]/modules/com/mycompany/main/`**.

5. Copy your jar file into the newly-created directory.

6. Create a module.xml file and also put it in the **`[JDV_HOME]/modules/com/mycompany/main/`** directory.

   Here is an example that shows dependencies specific to JBoss Data Virtualization:

```
<?xml version="1.0" encoding="UTF-8"?>
  <module xmlns="urn:jboss:module:1.0" name="com.mycompany">
         <resources>
            <resource-root path="my_custom_policy.jar" />
            <!--add any other dependent jars here, if they are not
defined as modules -->
        </resources>
    <dependencies>
        <module name="org.jboss.teiid.common-core"/>
        <module name="org.jboss.teiid.api"/>
        <module name="javax.api"/>
    </dependencies>
</module>
```

> **NOTE**
>
> If your PolicyDecider has any third-party dependencies, add them as dependencies to the **`module.xml`** file. Ensure you list all the files required. If dependencies are missing you will be informed when you start the software.

7. After the module has been added, edit the server configuration file (**`standalone.xml`** or its equivalent) to use your module name. The module must be added to the "teiid" subsystem:

```
<policy-decider-module>MODULE-NAME</policy-decider-module>
```

8. Restart the system.

CHAPTER 6. CONFIGURING TRANSPORT SECURITY

6.1. TRANSPORT SECURITY CONFIGURATION

By default, JBoss Data Virtualization provides two types of remote transports, each with its own SSL configuration: **`teiid`** and **`pg`**. The **`teiid`** transport encrypts login traffic only and the **`pg`** transport defaults to no SSL.

NOTE

The **`pg`** transport for ODBC access defaults to cleartext username and password authentication.

The following is a more comprehensive example of SSL configuration as it would appear within the **`transport`** definition.

```
<ssl mode="enabled" authentication-mode="1-way" ssl-protocol="SSLv3"
keymanagement-algorithm="algo"
         enabled-cipher-
suites="SSL_RSA_WITH_RC4_128_MD5,SSL_RSA_WITH_RC4_128_SHA">
            <keystore name="cert.keystore" password="passwd" type="JKS"
key-alias="alias"/>
            <truststore name="cert.truststore" password="passwd"/>
</ssl>
```

6.2. TRANSPORT SECURITY PROPERTIES

The following properties can be set when defining the transport security setting for a transport.

Table 6.1. SSL Properties

Setting	Description	Default Value
mode	Options are: **`disabled`**, **`login`**, or **`enabled`**. If set to **`disabled`**, no transport or message level encryption will be used. If set to **`login`**, only the login traffic will be encrypted at a message level using 128 bit AES with an ephemeral DH key exchange. This only applies to the **`teiid`** transport. (No other configuration values are required in this mode.) If set to **`enabled`**, traffic will be encrpyted using SSL according to the configuration properties below. **`teiid`** transport clients must connect using SSL with the mms protocol. ODBC **`pg`** transport clients may optionally use SSL.	login

Setting	Description	Default Value
keystore/name	The filename of the keystore that contains the private key of the server. The file name can be specified relative to the JBoss Data Virtualization deployer classloader or by absolute file system path. A typical installation would place the keystore file in the ***EAP_HOME/MODE/*configuration** directory.	cert.keystore
keystore/password	The password used to access the keystore.	
keystore/type	The keystore type created by the keytool.	JKS
keystore/key-alias	The keystore key-alias created by the keytool.	
ssl-protocol	Type of SSL protocol to be used.	TLSv1
keymanagement-algorithm	Type of key algorithm to be used.	
truststore/name	If **authentication-mode** is set to **2-way**, this property must be provided. This is the truststore that contains the public key for the client. Depending on how you created the keystore and truststores, this may be the same as the file specified for **keystore/name**.	cert.truststore
truststore/password	The password used to access the truststore.	
authentication-mode	Options are **1-way**, **2-way** and **anonymous**.	1-way

Setting	Description	Default Value
enabled-cipher-suites	A comma separated list of cipher suites allowed for encryption between the client and server. The values must be supported by the JVM, otherwise the SSL connections will fail. **NOTE** Both anonymous SSL and login only encryption are configured to use 128 bit AES encryption by default. By default, 1-way and 2-way SSL allow for cipher suite negotiation based upon the default cipher suites supported by the respective Java platforms of the client and server. Administrators can restrict the cipher suites used for encryption by setting the **`enabled-cipher-suites`** property.	SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA, TLS_KRB5_WITH_RC4_128_MD5, TLS_KRB5_WITH_RC4_128_SHA, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_KRB5_WITH_3DES_EDE_CBC_MD5, TLS_KRB5_WITH_3DES_EDE_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_DSS_WITH_AES_256_CBC_SHA,

Setting	Description	Default Value
		TLS_RSA_WITH_AES_256_CBC_SHA

NOTE

You will typically use the CLI to modify the transport configuration.

WARNING

Red Hat recommends to encrypt passwords in production systems. Refer to the JBoss Enterprise Application Platform *Security Guide* for information about the *Password Vault*.

6.3. TRANSPORT SECURITY AUTHENTICATION MODES

The following authentication modes are available:

`anonymous`

No certificates are exchanged. Settings are not needed for the keystore and truststore properties. The client must have **`org.teiid.ssl.allowAnon`** set to true (the default) to connect to an anonymous server. Communications are encrypted using the TLS_DH_anon_WITH_AES_128_CBC_SHA SSL cipher suite. This is suitable for most secure intranets.

`1-way`

Athenticates the server to the client. The server presents a certificate which is signed by the private key stored in the server's keystore. The server's corresponding public key must be in the client's truststore.

`2-way`

Mutual client and server authentication. The server presents a certificate which is signed by the private key stored in the server's keystore. The server's corresponding public key must be in the client's truststore. Additionally, the client presents a certificate signed by its private key stored in the client's keystore. The client's corresponsing public key must be in the server's truststore.

NOTE

You can use keytool to generate encryption keys; however, you should first consider your local requirements for managing public key cryptography.

See Also:

- Section 9.1, “Keytool”

CHAPTER 7. DATA SOURCE SECURITY

7.1. DATA SOURCE SECURITY

Data source security is about security considerations when JBoss Data Virtualization connects to data sources it is integrating. The following are other security considerations not already covered.

Authorization

Any data source level authorization decisions are up to the source systems being integrated.

Encryption

Encrypting remote data source access is the responsibility for the resource adapter and library/driver used to access the source system.

Translators

The translator framework also provides hooks for securing access at the datasource level. ExecutionFactory.getConnection may be overridden to initialize the source connection in any number of ways, such as re-authentication, based upon the Subject, execution payload, session variables, and any of the other relevant information accessible via the ExecutionContext and the CommandContext. You may even also modify the generated source SQL in any way that is seen fit in the relevant Execution.

7.2. AUTHENTICATION MODULES FOR DATA SOURCE SECURITY

In some use cases, a user might need to supply credentials to data sources based on the logged in user, rather than shared credentials for all the logged users. To support this feature, JBoss EAP and JBoss Data Virtualization provide additional authentication modules to be used in conjunction with the main security domain:

- Caller Identity Login Module
- Role-Based Credential Map Identity Login Module

7.3. CALLER IDENTITY LOGIN MODULE

If a client needs to supply a simple text password, certificate, or a custom serialized object as a credential to the data source, administrators can configure the **`CallerIdentityLoginModule`**. Using this login module, users are able to supply to the data source the same credential used to log into the JBoss Data Virtualization security domain.

7.4. CONFIGURING THE CALLER IDENTITY LOGIN MODULE

Procedure 7.1. Configure the Caller Identity Login Module

1. **Create the Login Module**
   Configure authentication modules using the Management Console according to the following specification:

```
<security-domain name="my-security-domain" cache-type="default">
    <authentication>
```

```
        <login-module
code="org.picketbox.datasource.security.CallerIdentityLoginModule"
module="org.picketbox" flag="required">
            <module-option name="password-stacking"
value="useFirstPass"/>
            <module-option name="userName" value="guest"/>
            <module-option name="password" value="guest"/>
        </login-module>
    </authentication>
</security-domain>
```

2. - **Configure the Data Source**
     Configure the datasource according to the following specification.

     ```
     <datasource jndi-name="java:/mysql-ds" pool-name="mysql-ds"
     enabled="true">
         <connection-url>jdbc:mysql://localhost:3306/txns</connection-
     url>
         <driver>mysql</driver>
          <pool><allow-multiple-users/></pool>
          <security>
               <security-domain>my-security-domain</security-domain>
          </security>
     </datasource>
     ```

   - **Configure the Connection Factory**
     Configure the resource adapter according to the following specification:

     ```
     <resource-adapter>
                 <archive>teiid-connector-ldap.rar</archive>
                 <transaction-support>NoTransaction</transaction-
     support>
                 <connection-definitions>
                     <connection-definition class-
     name="org.teiid.resource.adapter.ldap.LDAPManagedConnectionFactor
     y"
                             jndi-name="java:/ldapDS"
                             enabled="true"
                             use-java-context="true"
                             pool-name="ldap-ds">

                       <config-property
     name="LdapUrl">ldap://ldapServer:389</config-property>
                       <config-property
     name="LdapAdminUserDN">cn=???,ou=???,dc=???</config-property>
                       <config-property
     name="LdapAdminUserPassword">pass</config-property>
                       <config-property
     name="LdapTxnTimeoutInMillis">-1</config-property>

                       <security>
                          <security-domain>my-security-
     domain</security-domain>
                      </security>
     ```

```
                  </connection-definition>
              </connection-definitions>
          </resource-adapter>
```

Result

When a user logs in with a password, the same password will also be set on the logged in Subject after authentication. These credentials can be extracted by the data source by asking for Subject's private credentials.

7.5. ROLE-BASED CREDENTIAL MAP IDENTITY LOGIN MODULE

In some cases, access to data sources is defined by roles, and users are assigned these roles, taking on the privileges that come with having them.

JBoss Data Virtualization provides a login module called **`RoleBasedCredentialMap`** for this purpose.

An administrator can define a role-based authentication module where, given the role of the user from the primary login module, this module will hold a credential for that role. Each role is associated with a set of credentials. If a user has multiple roles, the first role that has the required credential will be used.

7.6. CONFIGURING THE ROLE-BASED CREDENTIAL MAP IDENTITY LOGIN MODULE

Procedure 7.2. Configure Role-Based Credential Map Identity Login Module

1. **Create the Login Module**
   Configure authentication modules using the Management Console according to the following specification:

```
   <subsystem xmlns="urn:jboss:domain:security:1.1">
   </subsystem>
    <security-domains>
    </subsystem>
        <security-domain name="my-security-domain" cache-
type="default">
        </security-domain>
            <authentication>
                <login-module code="UsersRoles" flag="required">
                </subsystem>
                    <module-option name="password-stacking"
value="useFirstPass"/>
                    <module-option name="usersProperties"
value="file://${jboss.server.config.dir}/teiid-security-
users.properties"/>
          <module-option name="rolesProperties"
value="file://${jboss.server.config.dir}/teiid-security-
roles.properties"/>
                </login-module>

                <login-module
code="org.teiid.jboss.RoleBasedCredentialMapIdentityLoginModule"
flag="required">
                </login-module>
```

```
                    <module-option name="password-stacking"
value="useFirstPass"/>
                    </login-module>
                    <module-option name="credentialMap"
value="file://${jboss.server.config.dir}/teiid-
credentialmap.properties"/>
                </login-module>

            </authentication>
        </security-domain>
    </security-domains>
    </subsystem>
```

2. **Complete the Configuration**
   Configure the data source or connection factory in the same way as for the **`CallerIdentityLoginModule`**.

Result

In the above example, the primary login module **`UsersRolesLoginModule`** is configured to login the primary user and assign some roles. The **`RoleBasedCredentialMap`** login module is configured to hold role to password information in the file defined by the **`credentialMap`** property. When the user logs in, the role information from the primary login module is taken, and the role's password is extracted and attached as a private credential to the Subject.

NOTE

To use an encrypted password instead of a plaintext one, include the encrypted password in the file defined by the **`credentialMap`** property.

For more information about encrypting passwords, refer to the *JBoss Enterprise Application Platform Security Guide*.

CHAPTER 8. KERBEROS SUPPORT

8.1. KERBEROS SUPPORT USING GSSAPI

JBoss Data Virtualization supports kerberos authentication using GSSAPI, to be used with single sign-on applications. This service ticket negotiation based authentication is supported through remote JDBC and ODBC drivers and LocalConnections. Client configuration will differ depending on connection types.

8.2. KERBEROS SUPPORT: LOCAL CONNECTION

Procedure 8.1. Setup Kerberos Support for Local Connection

- Set the JDBC URL property **`PassthroughAuthentication`** to true and use JBoss Negotiation for authentication of your Web application with Kerberos.

Result

When the web application authenticates with the provided Kerberos token, the same authenticated subject will be used in JBoss Data Virtualization.

NOTE

For more information about JBoss Negotiation and SPNEGO, refer to the JBoss Enterprise Application Platform *Security Guide*.

8.3. KERBEROS SUPPORT: REMOTE CONNECTION

Procedure 8.2. Configure Kerberos for Remote Connection

1. **Define the security domain**
   Now we need to define a security domain for **`krb5-domain`**.

   Since Kerberos cannot define authorization roles, it relies on authorization provided by the **`teiid-security`** domain. The following example uses the **`UsersRolesLoginModule`** for this purpose.

   NOTE

   This configuration replaces the default JBoss Data Virtualization login configuration, and you should change the **`principal`** and **`keyTab`** locations accordingly.

```
  <security-domain name="krb5-domain">
    <authentication>
        <login-module code="Kerberos" flag="required">
            <module-option name="storeKey" value="true"/>
            <module-option name="useKeyTab" value="true"/>
            <module-option name="principal"
value="dv/my.host.com@EXAMPLE.COM"/>
            <module-option name="keyTab"
value="/path/to/krb5.keytab"/>
            <module-option name="doNotPrompt" value="true"/>
            <module-option name="debug" value="false"/>
        </login-module>
    </authentication>
</security-domain>
<security-domain name="EXAMPLE.COM">
   <authentication>
      <!-- Check the username and password -->
      <login-module code="SPNEGO"  flag="requisite">
         <module-option name="password-stacking"
value="useFirstPass"/>
         <module-option name="serverSecurityDomain" value="krb5-
domain"/>
      </login-module>
      <!-- Search for roles -->
      <login-module code="UserRoles" flag="required">
         <module-option name="password-stacking"
value="useFirstPass"/>
         <module-option name="usersProperties"
value="file://${jboss.server.config.dir}/spnego-users.properties"/>
         <module-option name="rolesProperties"
value="file://${jboss.server.config.dir}/spnego-roles.properties"/>
      </login-module>
   </authentication>
</security-domain>
```

2. You can add following combination VDB properties in the vdb.xml file to select or force the security-domain and authentication type:

```
<property name="security-domain" value="EXAMPLE.COM"/>
<property name="gss-pattern" value="{regex}" />
<property name="password-pattern" value="{regex}" />
<property name="authentication-type" value="GSS or USERPASSWORD" />
```

All the properties above are optional on a VDB. If you want to define VDB based security configuration "security-domain" property is required. If you want to enforce single authentication type use "authentication-type" property is required. If your security domain can support both GSS and USERPASSWORD, then you can define "gss-pattern" and "password-pattern" properties, and define a regular expression as the value. During the connection, these regular expressions are matched against the connecting user's name provided to select which authentication method user prefers. Here is a sample configuration:

```
<property name="security-domain" value="EXAMPLE.COM"/>
<property name="gss-pattern" value="logasgss" />
```

In this case, if you passed the "user=logasgss" in the connection string, then GSS authentication is selected as login authentication mechanism. If the user name does not match, then default transport's authentication method is selected. Alternatively, if you want choose USERPASSWORD like this:

```
<property name="security-domain" value="EXAMPLE.COM"/>
<property name="password-pattern" value="*-simple" />
```

-

If the user name is "mike-simple", then that user will be subjected to authenticate against USERPASSWORD based authentication domain. You can configure different security-domains for different VDBS. VDB authentication will no longer be dependent upon underlying transport. If you wish to use "GSS" all the time then use this configuration:

```
<property name="security-domain" value="EXAMPLE.COM"/>
<property name="authentication-type" value="GSS" />
```

3. **Edit JVM configuration**
   Edit the ***EAP_HOME*/bin/*MODE*.conf** file and add the following JVM options (changing the **realm** and **kdc** settings according to your environment):

```
JAVA_OPTS = "$JAVA_OPTS -Djava.security.krb5.realm=EXAMPLE.COM -
Djava.security.krb5.kdc=kerberos.example.com -
Djavax.security.auth.useSubjectCredsOnly=false"
```

   You can also add these properties to the **standalone.xml** file, right after the {extensions} segment:

```
<system-properties>
    <property name="java.security.krb5.conf"
value="/pth/to/krb5.conf"/>
    <property name="java.security.krb5.debug" value="false"/>
    <property name="javax.security.auth.useSubjectCredsOnly"
value="false"/>
</system-properties>
```

4. **Set the security domains**
   There are two ways to do this. You first option is to define one default authentication per transport:

```
<transport name="jdbc" protocol="teiid" socket-binding="teiid-
jdbc"/>
<authentication security-domain="EXAMPLE.COM" type="GSS"/>
</transport>
```

   Your second option is to secure only one virtual database:

```
<property name="security-domain" value="EXAMPLE.COM"/>
<property name="authentication-type" value="GSS" />
```

5. **Restart the server**
   Restart the server, ensuring there are no errors as it launches.

8.4. KERBEROS SUPPORT: JDBC CLIENT CONFIGURATION

Procedure 8.3. Setup Kerberos for JDBC Client

1. **Set JAAS configuration**

In the client VM the JAAS configuration for kerberos authentication needs to be written. Here is sample configuration file (**`client.conf`**):

```
    Client {
     com.sun.security.auth.module.Krb5LoginModule required
     useTicketCache=true
     storeKey=true
     useKeyTab=true
     keyTab="/path/to/krb5.keytab"
     doNotPrompt=false
     debug=false
     principal="user@EXAMPLE.COM";
};
```

2. **Set JVM configuration**
   Add the following JVM options to your client's startup script. Change **`realm`** and **`kdc`** settings according to your environment:

```
-Djava.security.krb5.realm=EXAMPLE.COM
-Djava.security.krb5.kdc=kerberos.example.com
-Djavax.security.auth.useSubjectCredsOnly=false
-Dsun.security.krb5.debug=false
-Djava.security.auth.login.config=/path/to/client.conf
```

   Or if you want to control the **`kdc`** and **`realm`** system wide, use below instead.

```
-Djava.security.krb5.conf=/path/to/krb5.conf (on Linux
/etc/krb5.conf)
-Djava.security.auth.login.config=/path/to/client.conf
-Djavax.security.auth.useSubjectCredsOnly=false
-Dsun.security.krb5.debug=false
```

3. **Set URL connection properties**
   Add the following URL connection properties to the JBoss Data Virtualization JDBC connection string:

```
authenticationType=KRB5;jaasName=Client;kerberosServicePrincipleName
=dv/my.host.com@EXAMPLE.COM
```

NOTE

There is no need to provide the username and password. When the application makes a JDBC connection, it will authenticate locally and use the same user credentials to negotiate a service token with the server and grant the connection. For more information on connection properties and how to configure data sources, see the JBoss Data Virtualization Platform *Development Guide*.

8.5. KERBEROS SUPPORT: ODBC CLIENT CONFIGURATION

Refer to PostgreSQL ODBC client documentation.

CHAPTER 9. KEYTOOL

9.1. KEYTOOL

Keytool is an encryption key and certificate management utility. It enables users to create and manage their own public/private key pairs and associated certificates for use in self-authentication, and also to cache public keys (in the form of certificates) belonging to other parties, for securing communication to those parties.

9.2. USING KEYTOOL WITH JBOSS DATA VIRTUALIZATION

When using the keytool to manage public key cryptography for JBoss Data Virtualization, use the following options:

- Set the alias to **`teiid`** using the **`-alias teiid`** option.
- Set the algorithm to **`RSA`** using the **`-keyslg RSA`** option.
- Set the validity period to **`365`** days using the **`-validity 365`** option.
- Set the store type to **`JKS`** using the **`-storetype JKS`** option.

9.3. CREATE A PRIVATE/PUBLIC KEY PAIR WITH KEYTOOL

Procedure 9.1. Create a Private/Public Key Pair with Keytool

1. Run the **`keytool -genkey -alias`** ***`ALIAS`*** **`-keyalg`** ***`ALGORITHM`*** **`-validity`** ***`DAYS`*** **`-keystore`** ***`server.keystore`*** **`-storetype`** ***`TYPE`*** command:

   ```
   keytool -genkey -alias teiid -keyalg RSA -validity 365 -keystore
   server.keystore -storetype JKS
   ```

2. If the specified keystore already exists, enter the existing password for that keystore, otherwise enter a new password:

   ```
   Enter keystore password: <password>
   ```

3. Answer the following questions when prompted:

   ```
   What is your first and last name?
   [Unknown]:  <user's name>
   What is the name of your organizational unit?
   [Unknown]:  <department name>
   What is the name of your organization?
   [Unknown]:  <company name>
   What is the name of your City or Locality?
   [Unknown]:  <city name>
   What is the name of your State or Province?
   [Unknown]:  <state name>
   What is the two-letter country code for this unit?
   [Unknown]:  <country name>
   ```

4. Enter **`yes`** to confirm the provided information is correct:

   ```
   Is CN=<user's name>, OU=<department name>, O="<company name>",
   L=<city name>, ST=<state name>, C=<country name>  correct?
   [no]:  yes
   ```

5. Enter your desired keystore password:

   ```
   Enter key password for <server>
   (Return if same as keystore password)
   ```

Result

The **`server.keystore`** file contains the newly generated public and private key pair.

9.4. EXTRACT A SELF-SIGNED CERTIFICATE FROM THE KEYSTORE

Procedure 9.2. Extract a Self-signed Certificate from the Keystore

1. Run the **`keytool -export -alias`** ***`ALIAS`*** **`-keystore`** ***`server.keystore`*** **`-rfc -file`** ***`public.cert`*** command:

   ```
   keytool -export -alias teiid -keystore server.keystore -rfc -file
   public.cert
   ```

2. Enter the keystore password when prompted:

   ```
   Enter keystore password: <password>
   ```

Result

This creates the **`public.cert`** file that contains a certificate signed with the private key in the **`server.keystore`**.

9.5. ADD A CERTIFICATE TO A TRUSTSTORE USING KEYTOOL

Procedure 9.3. Add a Certificate to a Truststore Using Keytool

1. Run the **`keytool -import -alias`** ***`ALIAS`*** **`-file`** ***`public.cert`*** **`-storetype`** ***`TYPE`*** **`-keystore`** ***`server.truststore`*** command:

   ```
   keytool -import -alias teiid -file public.cert -storetype JKS -
   keystore server.truststore
   ```

2. If the specified truststore already exists, enter the existing password for that truststore, otherwise enter a new password:

   ```
   Enter keystore password:  <password>
   ```

3. Enter **`yes`** when prompted to trust the certificate:

```
Owner: CN=<user's name>, OU=<dept name>, O=<company name>, L=<city>,
ST=<state>, C=<country>
Issuer: CN=<user's name>, OU=<dept name>, O=<company name>, L=
<city>, ST=<state>, C=<country>
Serial number: 416d8636
Valid from: Fri Jul 31 14:47:02 CDT 2009 until: Sat Jul 31 14:47:02
CDT 2010
Certificate fingerprints:
         MD5:  22:4C:A4:9D:2E:C8:CA:E8:81:5D:81:35:A1:84:78:2F
         SHA1:
05:FE:43:CC:EA:39:DC:1C:1E:40:26:45:B7:12:1C:B9:22:1E:64:63
Trust this certificate? [no]:  yes
```

Result

The certificate in **`public.cert`** has been added to the new truststore named **`server.truststore`**.

CHAPTER 10. SAML-BASED SECURITY FOR ODATA

10.1. SAML-BASED SECURITY FOR ODATA

By default, the OData access to a Virtual Database (VDB) in Red Hat JBoss EAP uses the HTTP Basic authentication method.

However, you can also configure OData to utilize Single-Sign-On (SSO)-based security using SAML2. This is how you do so with PicketLink.

Prerequisites

- An identity provider configued and working with the security domain of your choice such as LDAP or Kerberos. (In this example it is an OData WAR file.)
- You must have the certificate for authentication that is used by IDP to sign the SAML messages.
- The PicketLink subsystem must be installed in your EAP instance.
- The DNS names for the machines on which Red Hat JBoss EAP is installed.

  NOTE

  Do not try to use IP address or localhost except for in testing scenarios. Configure proper DNS names for both the IDP and the security provider servers and make sure both can access each other using the URLs you have configured.

- The SSO POST-based URL for your IDP, that your security provider can use to redirect for authentication call.

1. Add an extension to the PicketLink subsystem:

```
  <extensions>
  <extension module="org.picketlink.as.extension" />
<extensions>
```

2. Configure the PicketLink subsystem:

```
 <subsystem xmlns="urn:jboss:domain:picketlink:1.0">
  <federation alias="odata">
    <saml token-timeout="4000" clock-skew="0"/>
    <key-store url="/\{CERTIFICATE-FILE-NAME\}" passwd="\{PASSWD\}"
sign-key-alias="\{CERTIFICATE-ALIAS\}" sign-key-passwd="\
{PASSWD\}"/>
    <identity-provider url="\{SSO-IDP-POST-URL\}" alias="idp.war"
security-domain="idp" supportsSignatures="true" strict-post-
binding="true">

      <trust>
        <trust-domain name="localhost" cert-alias="\{CERTIFICATE-
ALIAS\}"/>
```

```
        <trust-domain name="127.0.0.1" cert-alias="\{CERTIFICATE-
ALIAS\}"/>
        <trust-domain name="{IDP-DNS-NAME}" cert-alias="\
{CERTIFICATE-ALIAS\}"/>
      </trust>
    </identity-provider>
    <service-providers>
    <service-provider alias="odata.war" security-domain="sp"
url="http://\{SP-DNS-NAME\}:8080/odata/" post-binding="true"
supportsSignatures="true"/>
    </service-providers>
  </federation>
</subsystem>
```

NOTE

Normally, the certificate alias is the domain name, such as "idp.jboss.org"

3. Configure the security domains to be used by the security provider:

```
 <subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="sp" cache-type="default">
            <authentication>
                <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2Lo
ginModule" flag="required"/>
                <login-module
code="org.jboss.security.ClientLoginModule" flag="required"/>
            </authentication>
        </security-domain>
    </security-domains>
</subsystem>
```

4. Change the OData transport in the Teiid subsystem:

```
  <transport name="odata">
     <authentication security-domain="sp"/>
   </transport>
```

5. Move the **teiid-odata-xxxx.war** file from **dataVirtualization/vdb/teiid-odata-xxx.war** to a temporary location.

6. Edit the **jboss-web.xml** file:

```
 <?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
    <context-root>odata</context-root>
</jboss-web>
```

7. Edit the **`web.xml`** file:

```
 <?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name>odata</display-name>
    <context-param>
        <param-name>javax.ws.rs.Application</param-name>
        <param-value>org.teiid.odata.TeiidODataApplication</param-
value>
    </context-param>
    <context-param>
        <param-name>batch-size</param-name>
        <param-value>256</param-value>
    </context-param>
    <context-param>
        <param-name>skiptoken-cache-time</param-name>
        <param-value>300000</param-value>
    </context-param>
    <context-param>
        <param-name>local-transport-name</param-name>
        <param-value>odata</param-value>
    </context-param>
    <listener>
        <listener-
class>org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap</l
istener-class>
    </listener>
    <servlet>
        <servlet-name>Resteasy</servlet-name>
        <servlet-class>org.teiid.odata.ODataServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>Resteasy</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <security-constraint>
        <display-name>require valid user</display-name>
        <web-resource-collection>
            <web-resource-name>Teiid Rest Application</web-resource-
name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>*</role-name>
        </auth-constraint>
    </security-constraint>

    <login-config>
        <auth-method>FORM</auth-method>
```

```
            <realm-name>sp</realm-name>
            <form-login-config>
                <form-login-page>/jsp/login.jsp</form-login-page>
                <form-error-page>/jsp/loginerror.jsp</form-error-page>
            </form-login-config>
        </login-config>

         <security-role>
            <description>security role</description>
            <role-name>*</role-name>
        </security-role>

    </web-app>
```

8. Add the certificate received from IDP vendor to the **`WEB-INF/classes`** directory.

   NOTE

   This must be same name as {CERTIFICATE-FILE-NAME} used in when you configured the PicketLink subsystem.

9. Recreate the WAR file based on the modified contents of the other files by running this command: **`jar -cvf teiid-odata-xxxx.war /temp/*`**

10. Copy the newly-created WAR file into the **`/modules/system/base/org/jboss/teiid/main/deployments`** directory.

11. Start the Data Virtualization server, and access the OData URL, You will be redirected to the SSO-based authentication.

CHAPTER 11. PATCH INSTALLATION

11.1. ABOUT PATCHES AND UPGRADES

The patching mechanism in JBoss EAP 6 applies updates which are made available to a specific 'minor' version of JBoss EAP 6, for example JBoss EAP 6.2. Patches can contain one-off, security, or cumulative updates.

Upgrading between major and minor releases of JBoss EAP (for example, from 6.1 to 6.2) requires a different process.

11.2. ABOUT PATCHING MECHANISMS

JBoss patches are released in two forms.

- Asynchronous updates: one-off patches which are released outside the normal update cycle of the existing product. These may include security patches, as well as other one-off patches provided by Red Hat Global Support Services (GSS) to fix specific issues.

- Planned updates: These include cumulative patches, as well as micro, minor or major upgrades of an existing product. Cumulative patches include all previously developed asynchronous updates for that version of the product.

Deciding whether a patch is released as part of a planned update or an asynchronous update depends on the severity of the issue being fixed. An issue of low impact is typically deferred, and is resolved in the next cumulative patch or minor release of the affected product. Issues of moderate or higher impact are typically addressed in order of importance as an asynchronous update to the affected product, and contain a fix for only a specific issue.

Cumulative and security patches for JBoss products are distributed in two forms: zip (for all products) and RPM (for a subset of products).

IMPORTANT

A JBoss product installation must always only be updated using one patch method: either zip or RPM patches.

Security updates for JBoss products are provided by an erratum (for both zip and RPM methods). The erratum encapsulates a list of the resolved flaws, their severity ratings, the affected products, textual description of the flaws, and a reference to the patches. Bug fix updates are not announced via an erratum.

For more information on how Red Hat rates JBoss security flaws, refer to: Section 11.6, “Severity and Impact Rating of JBoss Security Patches”

Red Hat maintains a mailing list for notifying subscribers about security related flaws. See Section 11.3, “Subscribe to Patch Mailing Lists”

11.3. SUBSCRIBE TO PATCH MAILING LISTS

Summary

The JBoss team at Red Hat maintains a mailing list for security announcements for Red Hat JBoss Enterprise Middleware products. This topic covers what you need to do to subscribe to this list.

Prerequisites

- None

Procedure 11.1. Subscribe to the JBoss Watch List

1. Click the following link to go to the JBoss Watch mailing list page: JBoss Watch Mailing List.
2. Enter your email address in the **`Subscribing to Jboss-watch-list`** section.
3. [You may also wish to enter your name and select a password. Doing so is optional but recommended.]
4. Press the **`Subscribe`** button to start the subscription process.
5. You can browse the archives of the mailing list by going to: JBoss Watch Mailing List Archives.

Result

After confirmation of your email address, you will be subscribed to receive security related announcements from the JBoss patch mailing list.

11.4. INSTALL PATCHES IN ZIP FORM

11.4.1. The `patch` Command

The **`patch`** command is used to apply downloaded zip patches to a single JBoss EAP 6 server instance. It cannot be used to automatically patch JBoss EAP 6 server instances across a managed domain, but individual server instances in a managed domain can be patched independently.

IMPORTANT

JBoss EAP 6 server instances which have been installed using the RPM method cannot be updated using the **`patch`** command. Refer to Section 11.5, “Install Patches in RPM form” to update RPM-installed JBoss EAP 6 servers.

NOTE

The **`patch`** command can only be used with patches produced for versions of JBoss EAP 6.2 and later. For patches for versions of JBoss EAP prior to 6.2, you should instead refer to the relevant version's documentation available at https://access.redhat.com/site/documentation/.

In addition to applying patches, the **`patch`** command can give basic information on the state of installed patches, and also provides a way to immediately rollback the application of a patch.

Before starting a patch application or rollback operation, the **`patch`** tool will check the modules and other miscellaneous files that it is changing for any user modifications. If a user modification is detected, and a conflict-handling switch has not been specified, the **`patch`** tool will abort the operation and warn that there is a conflict. The warning will include a list of the modules and other files that are in conflict. To complete the operation, the **`patch`** command must be re-run with a switch specifying how to resolve the conflict: either to preserve the user modifications, or to override them.

Table 11.1. `patch` Command Arguments and Switches

Argument or Switch	Description
`apply`	Applies a patch.
`--override-all`	If there is a conflict, the patch operation overrides any user modifications.
`--override-modules`	If there is a conflict as a result of any modified modules, this switch overrides those modifications with the contents of the patch operation.
`--override=`***`path`***`(,`***`path`***`)`	For specified miscellaneous files only, this will override the conflicting modified files with the files in the patch operation.
`--preserve=`***`path`***`(,`***`path`***`)`	For specified miscellaneous files only, this will preserve the conflicting modified files.
`info`	Returns information on currently installed patches.
`rollback`	Rollsback the application of a patch.
`--reset-configuration=`***`TRUE`***`|`***`FALSE`***	Required for rollback, this specifies whether to restore the server configuration files as part of the rollback operation.

11.4.2. Installing Patches in Zip Form Using the `patch` Command

Summary

This task describes how to use the **`patch`** command to install patches for JBoss EAP 6 that are in the zip format.

IMPORTANT

The **`patch`** command is a feature that was added in JBoss EAP 6.2. For versions of JBoss EAP prior to 6.2, the process to install patches in zip form is different, and you should instead refer to the relevant version's documentation available at https://access.redhat.com/site/documentation/.

Prerequisites

- Valid access and subscription to the Red Hat Customer Portal.
- A current subscription to a JBoss product installed in zip format.
- Access to the Management CLI for the server instance to be updated. Refer to *Launch the Management CLI* in the *Administration and Configuration Guide*.

Procedure 11.2. Apply a zip patch to a JBoss EAP 6 server instance using the`patch` command

WARNING

Before installing a patch, you should backup your JBoss product along with all customized configuration files.

1. Download the patch zip file from the Customer Portal at https://access.redhat.com/downloads/
2. From the Management CLI, apply the patch with the following command with the appropriate path to the patch file:

   ```
   [standalone@localhost:9999 /] patch apply /path/to/downloaded-patch.zip
   ```

   The **`patch`** tool will warn if there are any conflicts in attempting the apply the patch. Refer to Section 11.4.1, "The **`patch`** Command" for available switches to re-run the command to resolve any conflicts.
3. Restart the JBoss EAP 6 server instance for the patch to take effect:

   ```
   [standalone@localhost:9999 /] shutdown --restart=true
   ```

Result

The JBoss EAP 6 server instance is patched with the latest update.

11.4.3. Rollback the Application of a Patch in Zip Form Using the `patch` Command

Summary

This task describes how to use the **`patch`** command to rollback the application of a previously applied zip patch in JBoss EAP 6.

WARNING

Rolling back the application of a patch using the **`patch`** command is not intended as a general uninstall functionality. It is only intended to be used immediately after the application of a patch which had undesirable consequences.

IMPORTANT

The **`patch`** command is a feature that was added in JBoss EAP 6.2. For versions of JBoss EAP prior to 6.2, the process to rollback patches in zip form is different, and you should instead refer to the relevant version's documentation available at https://access.redhat.com/site/documentation/.

Prerequisites

- A patch that was previously applied using the **`patch`** command.
- Access to the Management CLI for the server instance. Refer to *Launch the Management CLI* in the *Administration and Configuration Guide*.

Procedure 11.3. Rollback a patch from a JBoss EAP 6 server instance using the `patch` command

1. From the Management CLI, use the **`patch info`** command to find the ID of the patch that is to be rolled back.
   - For cumulative patches, the patch ID is the value of the first **`cumulative-patch-id`** shown in the **`patch info`** output.
   - One-off security or bug fix patch IDs are listed as the value of the first **`patches`** shown in the **`patch info`** output, with the most recently applied one-off patch listed first.
2. From the Management CLI, rollback the patch with the appropriate patch ID from the previous step.

   WARNING

   Use caution when specifying the value of the **`--reset-configuration`** switch.

   If set to **`TRUE`**, the patch rollback process will also rollback the JBoss EAP 6 server configuration files to their pre-patch state. Any changes that were made to the JBoss EAP 6 server configuration files after the patch was applied will be lost.

   If set to **`FALSE`**, the server configuration files will not be rolled back. In this situation, it is possible that the server will not start after the rollback, as the patch may have altered configurations, such as namespaces, which may no longer be valid and have to be fixed manually.

   ```
   [standalone@localhost:9999 /] patch rollback PATCH_ID --reset-
   configuration=TRUE
   ```

   The **`patch`** tool will warn if there are any conflicts in attempting the rollback the patch. Refer to Section 11.4.1, “The **`patch`** Command” for available switches to re-run the command to resolve any conflicts.
3. Restart the JBoss EAP 6 server instance for the patch rollback to take effect:

   ```
   [standalone@localhost:9999 /] shutdown --restart=true
   ```

Result

The patch, and optionally also the server configuration files, are rolled back on the JBoss EAP 6 server instance.

11.5. INSTALL PATCHES IN RPM FORM

Summary

JBoss patches are distributed in two forms: ZIP (for all products) and RPM (for a subset of products). This task describes the steps you need to take to install the patches via the RPM format.

Prerequisites

- A valid subscription to the Red Hat Network.
- A current subscription to a JBoss product installed via an RPM package.

Procedure 11.4. Apply a patch to a JBoss product via the RPM method

Security updates for JBoss products are provided by errata (for both zip and RPM methods). The errata encapsulates a list of the resolved flaws, their severity ratings, the affected products, textual description of the flaws, and a reference to the patches.

For RPM distributions of JBoss products, the errata include references to the updated RPM packages. The patch can be installed by using **yum**.

WARNING

Before installing a patch, you must backup your JBoss product along with all customized configuration files.

1. Get notified about the security patch either via being a subscriber to the JBoss watch mailing list or by browsing the JBoss watch mailing list archives.

2. Read the errata for the security patch and confirm that it applies to a JBoss product in your environment.

3. If the security patch applies to a JBoss product in your environment, then follow the link to download the updated RPM package which is included in the errata.

4. Use

   ```
   yum update
   ```

   to install the patch.

   IMPORTANT

   When updating an RPM installation, your JBoss product is updated cumulatively with all RPM-released fixes.

Result

The JBoss product is patched with the latest update using the RPM format.

11.6. SEVERITY AND IMPACT RATING OF JBOSS SECURITY PATCHES

To communicate the risk of each JBoss security flaw, Red Hat uses a four-point severity scale of low, moderate, important and critical, in addition to Common Vulnerability Scoring System (CVSS) version 2 base scores which can be used to identify the impact of the flaw.

Table 11.2. Severity Ratings of JBoss Security Patches

Severity	Description
Critical	This rating is given to flaws that could be easily exploited by a remote unauthenticated attacker and lead to system compromise (arbitrary code execution) without requiring user interaction. These are the types of vulnerabilities that can be exploited by worms. Flaws that require an authenticated remote user, a local user, or an unlikely configuration are not classed as critical impact.
Important	This rating is given to flaws that can easily compromise the confidentiality, integrity, or availability of resources. These are the types of vulnerabilities that allow local users to gain privileges, allow unauthenticated remote users to view resources that should otherwise be protected by authentication, allow authenticated remote users to execute arbitrary code, or allow local or remote users to cause a denial of service.
Moderate	This rating is given to flaws that may be more difficult to exploit but could still lead to some compromise of the confidentiality, integrity, or availability of resources, under certain circumstances. These are the types of vulnerabilities that could have had a critical impact or important impact but are less easily exploited based on a technical evaluation of the flaw, or affect unlikely configurations.
Low	This rating is given to all other issues that have a security impact. These are the types of vulnerabilities that are believed to require unlikely circumstances to be able to be exploited, or where a successful exploit would give minimal consequences.

The impact component of a CVSS v2 score is based on a combined assessment of three potential impacts: Confidentiality (C), Integrity (I) and Availability (A). Each of these can be rated as None (N), Partial (P) or Complete (C).

Because the JBoss server process runs as an unprivileged user and is isolated from the host operating system, JBoss security flaws are only rated as having impacts of either None (N) or Partial (P).

Example 11.1. CVSS v2 Impact Score

The example below shows a CVSS v2 impact score, where exploiting the flaw would have no impact

> on system confidentiality, partial impact on system integrity and complete impact on system availability (that is, the system would become completely unavailable for any use, for example, via a kernel crash).
>
> ```
> C:N/I:P/A:C
> ```

Combined with the severity rating and the CVSS score, organizations can make informed decisions on the risk each issue places on their unique environment and schedule upgrades accordingly.

For more information about CVSS2, please see: CVSS2 Guide.

APPENDIX A. REVISION HISTORY

Revision 6.2.0-23	**Tue Aug 4 2015**	**David Le Sage**

Updates for 6.2.