Red Hat JBoss Data Virtualization 6.3

Security Guide

This guide is intended for administrators

Red Hat Customer Content
Services

Red Hat JBoss Data Virtualization 6.3 Security Guide

This guide is intended for administrators

Red Hat Customer Content Services

Legal Notice

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.

> **Note**
>
> You should only select a profile if you are running Red Hat JBoss Data Virtualizaton in domain mode.

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#AppendixA. 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. On a standalone server, you can set system properties in the **System Properties** 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#AppendixA. 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 stored as a hash.

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 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.

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
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.	This defaults to all supported cipher suites for the virtual machine.

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.

> **Important**
>
> Anonymous SSL fails to work if you are using IBM's JDK.

`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 8.1, “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. A. **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>
```

   B. **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.LDAPManagedConnectionFactory"
                        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

> **Warning**
>
> RoleBasedCredentialMap is now deprecated.

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

> **Warning**
>
> RoleBasedCredentialMap is now deprecated.

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">
    <security-domains>
        <security-domain name="my-security-domain" cache-
type="default">
            <authentication>
                <login-module code="UsersRoles" flag="required">
                    <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://${jboss.server.config.dir}/teiid-security-
roles.properties"/>
                </login-module>
                <login-module
code="org.teiid.jboss.RoleBasedCredentialMapIdentityLoginModule"
flag="required">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                    <module-option name="credentialMap"
value="file://${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. Keytool

8.1. 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.

8.2. 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 algorithm to **RSA** using the **-keyslg RSA** option.
- Set the store type to **JKS** using the **-storetype JKS** option.

8.3. Create a Private/Public Key Pair with Keytool

Procedure 8.1. 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.

8.4. Extract a Self-signed Certificate from the Keystore

Procedure 8.2. 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**.

8.5. Add a Certificate to a Truststore Using Keytool

Procedure 8.3. 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 9. Kerberos Support Through GSS API

9.1. Kerberos and Red Hat JBoss Data Virtualization

9.1.1. Introduction to Kerberos Authentication on Red Hat JBoss Data Virtualization

Red Hat JBoss Data Virtualization supports Kerberos authentication using the GSS API for single sign-on applications. This service ticket negotiation-based authentication is supported through remote JDBC/ODBC drivers and LocalConnections. The client has to be configured differently for each variant.

9.1.1.1. Local Connection Overview

For a local connection, set the JDBC URL property PassthroughAuthentication to true and use JBoss Negotiation to authenticate your web-application with Kerberos. When the web application is authenticated with the provided Kerberos token, it can be used in Red Hat JBoss Data Virtualization. For details about how to configure this, please refer to the JBoss Negotiation documentation.

9.1.1.2. Remote Connections (JDBC/ODBC)

Open the standalone.xml file in your text editor.

Go to the "security-domains" section and add the following, customizing where necessary for your system.

> **Note**
>
> You need to configure two separate security domains. Configure one security domain to represent the identity of the server. The first security domain authenticates the container itself to the directory service. It needs to use a login module which accepts some type of static login mechanism, because a real user is not involved. This example uses a static principal and references a keytab file which contains the credential.

```
    <security-domain name="host">
       <authentication>
          <login-module code="Kerberos" flag="required">
             <module-option name="storeKey" value="true"/>
             <module-option name="useKeyTab" value="true"/>
             <module-option name="principal"
value="host/testserver@MY_REALM"/> <!-- service principal -->
             <module-option name="keyTab" value="/path/to/service.keytab"/>
             <module-option name="doNotPrompt" value="true"/>
             <module-option name="debug" value="false"/>
          </login-module>
       </authentication>
    </security-domain>
```

Configure a second security domain to secure the application. The second security domain is used to authenticate the individual user to the Kerberos server. (You need at least one login module to authenticate the user, and another to search for the roles to apply to the user.) The following XML code shows an example SPNEGO security domain. It includes an authorization module to map roles to individual users. You can also use a module which searches for the roles on the authentication server itself.

Note

The name of security-domain must match that of the realm.

```
<security-domain name="MY_REALM">
   <authentication>
      <!-- Check the username and password -->
      <login-module code="SPNEGO"  flag="requisite">
         <module-option name="password-stacking" value="useFirstPass"/>
         <module-option name="serverSecurityDomain" value="host"/>
      </login-module>
      <!-- Search for roles -->
      <login-module code="UserRoles" flag="requisite">
         <module-option name="password-stacking" value="useFirstPass" />
         <module-option name="usersProperties" value="spnego-
users.properties" />
         <module-option name="rolesProperties" value="spnego-
roles.properties" />
      </login-module>
   </authentication>
</security-domain>
```

9.1.1.3. User Roles and Groups

Kerberos does not assign any user roles to the authenticated subject. Therefore, you need to configure a separate role-mapping module to do this work. In the example above, the "UserRoles" login-module was added.

To assign groups, you must edit the "spnego-roles.properties" file and add them using this syntax: user@MY_REALM=my-group

Please refer to the Red Hat JBoss EAP documentation for more information about how to do this.

The SPENGO security-domain delegates the calls relating to Kerberos to the Kerberos server based on the "serverSecurityDomain" property.

To customise it, add the following to the SPENGO security domain:

```
 <module-option name="usernamePasswordDomain" value="{user-name-based-
auth}"/>
```

Once your security domains have been defined, you need to associate them with Red Hat JBoss Data Virtualization’s transport configuration or virtual database configuration. To define a default JDBC transport configuration, add this code:

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

For an ODBC transport, add this code:

```
<transport name="odbc" protocol="pg" socket-binding="teiid-odbc"/>
        <authentication security-domain="MY_REALM" type="GSS"/>
</transport>
```

Table 9.1. Type Values

Value	Description
USERPASSWORD	This only allows you to create username and password-based authentication.
GSS	This allows you to create GSS API-based authentications using Kerberos5.

To define a VDB-based authentication, add a combination of the optional following properties to the vdb.xml file:

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

Table 9.2. Table Properties

Property	Description
security-domain	Use this to define VDB-based security.
authentication-type	This allows you to enforce single authentication.
gss-pattern	This allows you to use GSS.
password-pattern	This allows you to use USERPASSWORD.

During the connection, these regular expressions are matched against the connecting user’s name to the user’s preferred authentication method. Here is an example:

```
<property name="security-domain" value="MY_REALM" />
<property name="gss-pattern" value="logasgss" />
```

In this case, if "user=logasgss" is passed in the connection string, then GSS authentication will be used to authenticate the user. If there is no match, then the default transport’s authentication method is selected.

You can configure different security-domains for different virtual databases and authentication will no longer be dependent upon the underlying transport.

For instance, if you wish make GSS the permanent default, use this code:

```
<property name="security-domain" value="MY_REALM" />
<property name="authentication-type" value="GSS" />
```

Open the {jboss-as}/bin/standalone.conf file in your text editor 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"
```

Alternatively, you can use this.

```
JAVA_OPTS = "$JAVA_OPTS -Djava.security.krb5.conf=/path/to/krb5.conf -
Djava.security.krb5.debug=false -
Djavax.security.auth.useSubjectCredsOnly=false"
```

Another way of doing this is to add these properties to the standalone.xml file, after the extensions section:

```
<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>
```

Restart the server. There should be no errors.

9.1.1.4. JDBC Client Configuration

You must configure your JDBC Client workstation so that it authenticates using the GSS API.

The workstation on which the JDBC Client exists must have been authenticated using GSS API against Active Directory or Enterprise directory server. Go to this website for information on this: http://spnego.sourceforge.net

You must now add a JAAS configuration for Kerberos authentication to your virtual machine. Here is a sample client.conf file:

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

Check that you have configured the "keytab" properly. For information on how to do this for Microsoft Windows environments, go to this website: http://spnego.sourceforge.net

For information on how to do this for Red Hat Enterprise Linux go to this site: https://access.redhat.com/site/solutions/208173

Add the following JVM options to your client’s initialization script, customizing the realm and KDC information for your environment This first sample is based on the krb5.conf file:

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

This alternative version is based on the KDC and Realm file:

```
-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
```

Add the following additional URL connection properties to the Red Hat JBoss Data Virtualization JDBC connection string along with the URL property:

```
jaasName=Teiid;user=
{pattern};kerberosServicePrincipleName=host/testserver@MY_REALM
```

Note

When you configure it to use Kerberos, you need to configure the "user" property as required by the "gss-pattern" or define the "authentication-type" property on the VDB or transport. However, after successful login into security-domain, the user name from the GSS login context is used instead.

Table 9.3. Properties

Value	Description
jaasName	This defines the JAAS configuration name in the client.conf file's java.security.auth.login.config property. This property is optional. If it is omitted, "Teiid" is used by default.
kerberosServicePrincipleName	This defines service principle that is requested on behalf of the service to which you are connecting. If this property is omitted, the default principle used is TEIID/hostname" and hostname is derived from the JDBC connection URL.

9.1.1.5. ODBC Client Configuration

Create a DSN for the virtual database on the client machine using the PostgreSQL ODBC driver. In order to participate in Kerberos based authentication you need to configure the "user" property as required by "gss-pattern" or define the "authentication-type" property on the VDB or transport.

No additional configuration is needed as part of this, except that your workstation where the ODBC DSN exists must have been authenticated using GSS API against Active Directory or other Enterprise directory server. For more details on this, see http://spnego.sourceforge.net

9.1.1.6. OData Client

By default, the OData client is configured to use HTTP Basic authentication.

To convert this authentication method into kerberos, clone or copy the maven project from https://github.com/teiid/teiid-web-security. You must update this to the version used in the product (featuring the redhat-x extension) before running Maven.

Edit the web.xml and jboss-web.xml files and then replace the MY_REALM property with that of your security domain.

Once the properties are updated, create a WAR file by running this command:

```
mvn clean install
```

Copy the WAR file from the odata-kerberos/target directory to replace the original OData WAR file with same name.

Chapter 10. OAuth2-Based Security for OData

10.1. OAuth2-Based Security for OData

In this example you will learn how to secure Red Hat JBoss Data Virtualization’s OData REST interface with OAuth2 using Red Hat SSO as the identity provider. It is also possible to delegate the negotiated OAuth2 access token at the OData interface, to the underlying data sources used in the Virtual Database, if the data source is also secured through Red Hat SSO as identity provider

10.1.1. Configuring Your Red Hat SSO Server

Prerequisites

You must have Red Hat SSO installed as a separate web server preferably on a different server machine. Please consult the Red Hat SSO documentation for more information about this.

1. Log into the Dashboard at http://localhost:8080/auth/admin/master/console/#/realms/oauth using the default "admin/admin" credentials. (You may have to create a Red Hat SSO administrative account in order to do this.)

2. Under the General tab, add a new realm called "oauth-demo".

3. Under Manage - Users, click "Add User" and add a new user called "user" with credentials.

4. Under Configure - Roles - Realm, click "Add Roles" and add the "odata" and "user" roles to your new user.

5. Under Configure - Clients - Settings, add a new client called "odata4-oauth". Add the odata and user roles and then choose scopes "odata" and "user" for this client.

> **Note**
>
> The redirect URI needs to be where the actual service is going to be available. Here is an example: http://[host]:[port]/odata4/*

The client web-service defines the roles the logged-in user must have in order to be granted access. In the Red Hat SSO OAuth2 implementation, these roles are used as "scopes". Note that the "odata4-oauth" client must have the scopes that it is going to delegate the access-token for gaining access to bottom data services. In this example Red Hat JBoss Data Virtualization’s OData web services requires the "odata" role. If you are delegating the access-token to the underlying web-service requires the "user" role.

> **Note**
>
> The "user" role is suggested as an example role required for the underlying source webservice. You need to replace it with the real role required for the webservice.

10.1.2. Configure the Red Hat JBoss Data Virtualization server

Prerequisites

Red Hat JBoss Data Virtualization server installed.

Red Hat SSO Adapter for EAP installed in Red Hat JBoss Data Virtualization.

1. Follow the instructions found here in the Red Hat SSO documentation to install the EAP 6 adapter: https://access.redhat.com/documentation/en/red-hat-single-sign-on/7.0/paged/securing-applications-and-services-guide/chapter-2-openid-connect

2. Change the OData transport's "security-domain" section using this CLI script:

```
/subsystem=teiid/transport=odata:write-attribute(name=authentication-
security-domain, value=oauth)
```

   The code will change to look like this:

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

   Note that, for this to deploy, you must also have the security domain configured on the server. To do so, run these CLI commands to create oauth and keycloack security domains:

```
/subsystem=security/security-domain=oauth:add(cache-type=default)
/subsystem=security/security-domain=oauth/authentication=classic:add
/subsystem=security/security-
domain=oauth/authentication=classic/login-
module=oauth:add(code=org.teiid.jboss.PassthroughIdentityLoginModule,
flag=required, module=org.jboss.teiid)

/subsystem=security/security-domain=keycloak:add(cache-type=default)
/subsystem=security/security-
domain=keycloak/authentication=classic:add
/subsystem=security/security-
domain=keycloak/authentication=classic/login-
module=keycloak:add(code=org.keycloak.adapters.jboss.KeycloakLoginModu
le, flag=required)

reload
```

3. Run the following CLI to undeploy the WAR file:

```
undeploy teiid-olingo-odata4.war
```

4. Download the Maven project found here: https://github.com/teiid/teiid-web-security

5. Replace the "teiid-web-security/teiid-odata-oauth-keycloak/src/main/webapp/WEB-INF/keyclock.json" file's contents with the "installation" script from the Red Hat SSO admin console's "odata4-client" client application.

6. Run this maven command to build the authentication module:

```
mvn clean package
```

7. Deploy the WAR file using this script:

```
deploy PATH/teiid-olingo-odata4.war
```

8. To **`test`** using a browser, go to this address: http://localhost:8180/odata4/keycloak/auth

9. Confirm that you can see a message that states "Congratulations!!! Login successful..".

   You can now execute any OData specific URL and that will participate in OAuth2 using OpenID Connect extension in RH-SSO.

Note

Testing only: When this testing is used, the token will expire after a time, meaning that you will need to negotiate for a new one by re-executing http://localhost:8180/odata4/keycloak/auth.

Note

If you are writing browser-based applications which will use this virtual database as a service to access data, these applications need mechanisms to negotiate access tokens from RH-SSO directly. The method discussed above is provided for testing purposes only.

You can also invoke this service programmatically. This process does not need to involve a web-browser and is typical of the situation whereby another web-application or mobile application is calling Red Hat JBoss Data Virtualization’s OData web-service to retrieve the data. The process of negotiating the "access-token" is externalized and is defined by the identity provider which, in this case, is Red Hat SSO.

To demonstrate, CURL is used below to negotiate this token (client_secret can found the Keycloak admin console under client credentials tab): curl -v POST http://localhost:8080/auth/realms/oauth-demo/protocol/openid-connect/token -H "Content-Type: application/x-www-form-urlencoded" -d 'username=user' -d 'password=user' -d 'grant_type=password' -d 'client_id=odata4-oauth' -d 'client_secret=36fdc2b9-d2d3-48df-8eea-99c0e729f525'

A JSON payload is returned:

```
     {  "access_token":"eyJhbGciOiJSUzI1NiJ9.eyJqdGkiOiI0YjI4NDMzYS1..",
        "expires_in":300,
        "refresh_expires_in":1800,

  "refresh_token":"eyJhbGciOiJSUzI1NiJ9.eyJqdGkiOiJmY2JmNjY2ZC0xNzIwLTQwODQtOT
  BiMi0wMjg4ODdhNDkyZWYiLCJl..",
        "token_type":"bearer",

  "id_token":"eyJhbGciOiJSUzI1NiJ9.eyJqdGkiOiIwZjYyNDQ1MS1iNTE0LTQ5YjUtODZlNy1
  jNTI5MDU2OTI3ZDIiLCJleH..",
        "not-before-policy":0,
        "session-state":"6c8884e8-c5aa-4f7a-a3fe-9a7f6c32658c"
     }
```

Now take the "access_token" and issue a query like this to access the OData service:

```
     curl -k -H "Authorization: Bearer
eyJhbGciOiJSUzI1NiJ9.eyJqdGkiOiI0YjI4NDMzYS1.."
http://localhost:8180/odata4/oauthdemo/view/message
```

You will see the same response message returned that you saw using the browser above.

10.1.3. Delegation of an OAuth2 Token to a Data Source

In order to delegate the same OAuth2 access token negotiated at the OData layer, you need to use RH-SSO as your identity provider in developing this web service. There are few additional steps you need to take to enable Red Hat SSO as the identity provider for this service:

1. Log into the RH-SSO admin console.
2. Under the realm “oauth-demo”, and then Clients - Settings, add another client called "database-service" and set the scope to "user". Set the type to "Bearer".
3. Follow the instructions in RH-SSO to secure this web service with above realm “oauth-demo”
4. Deploy this war into your web server.
5. Create a “web service” resource adapter for this service, using “oauth” as the security-domain.
6. Use this resource adapter as source in developing your virtual database.

> **Note**
>
> The delegation will occur if the virtual database is accessed through the OData interface. JDBC-based access to this virtual database would fail.

Chapter 11. SAML-Based Security for OData

11.1. SAML-Based Security for OData using Red Hat SSO

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 for SAML using Red Hat SSO.

Prerequisites

- An installed Red Hat SSO web server. It should be configured to have a new realm called oauth-demo, a new user, two enterprise roles called "user" and "odata" and a new client called "odata4-saml" that will represent the Data Virtualization SAML client. Finally, you need to either import or generate a new SAML key and then export it. Please refer to the Red Hat SSO documentation for details on how to achieve these tasks: https://access.redhat.com/documentation/en/red-hat-single-sign-on/7.0/
- You must have an installed and configured Red Hat JBoss Data Virtualization server.
- You must have a Red Hat SSO SAML adapter for EAP 6 that has been installed in the Red Hat JBoss Data Virtualization instance.

1. Add the Red Hat SSO-specific modules to the server. Please refer to the Red Hat SSO documentation for details: https://access.redhat.com/documentation/en/red-hat-single-sign-on/7.0/
2. Run these CLI commands:

```
/subsystem=security/security-domain=keycloak:add(cache-type=default)
/subsystem=security/security-
domain=keycloak/authentication=classic:add
/subsystem=security/security-
domain=keycloak/authentication=classic/login-
module=keycloak:add(code=org.keycloak.adapters.jboss.KeycloakLoginModu
le, flag=required)

reload
```

   The commands add this XML to the security-domain section of the domain.xml or standalone.xml file:

```
<security-domain name="keycloak">
    <authentication>
        <login-module
code="org.keycloak.adapters.jboss.KeycloakLoginModule"
flag="required"/>
    </authentication>
</security-domain>
```

3. Run this CLI command:

```
/subsystem=teiid/transport=odata:write-attribute(name=authentication-
security-domain, value=keycloak)
```

The command changes the security domain of the odata transport section of the domain.xml or standalone.xml file:

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

4. In order to use SAML authentication, the OData WAR needs to be updated to make use of the SAML-based security domain.

5. Run this CLI command to undeploy the WAR:

   ```
   undeploy teiid-olingo-odata4.war
   ```

6. Download the Maven project templates found here: https://github.com/teiid/teiid-web-security

7. Two WAR files are downloaded. Choose the SAML one.

8. Replace the "teiid-web-security/teiid-odata-saml-keycloak/src/main/webapp/WEB-INF/keyclock.json" file's content with the "installation" script in "keycloak.json" format from the Red Hat SSO admin console's "odata4-saml" client application.

9. Similarly, replace the "teiid-web-security/teiid-odata-saml-keycloak/src/main/webapp/WEB-INF/keystore.jks" file with the keystore you exported earlier.

10. Build the "keycloak-saml.xml" file, and add in all the sections of "metadata" specific to your service. In this way, the service will know where IDP located and which service this represents and so forth.

11. Build the WAR files using maven by issuing this command: **`mvn clean package`**

    A new WAR file is generated.

12. Run this script in the CLI to update the WAR file:

    ```
    undeploy teiid-olingo-odata4.war
    deploy teiid-web-security/teiid-odata-saml-keycloak/target/teiid-
    odata-saml-keycloak-{version}.war
    ```

    An alternative way to do this is to overlay the new WAR using this CLI script:

    ```
    deployment-overlay add --name=myOverlay --content=/WEB-
    INF/web.xml=teiid-web-security/teiid-odata-saml-
    keycloak/src/main/webapp/WEB-INF/web.xml,/WEB-INF/jboss-web.xml=teiid-
    web-security/teiid-odata-saml-keycloak/src/main/webapp/WEB-INF/jboss-
    web.xml,/META-INF/MANIFEST.MF=teiid-web-security/teiid-odata-saml-
    keycloak/src/main/webapp/META-INF/MANIFEST.MF,/WEB-INF/keycloak-
    saml.xml=teiid-web-security/teiid-odata-saml-
    keycloak/src/main/webapp/WEB-INF/keycloak-saml.xml,/WEB-
    INF/keycloak.jks=teiid-web-security/teiid-odata-saml-
    keycloak/src/main/webapp/WEB-INF/keycloak.jks --deployments=teiid-
    olingo-odata4.war --redeploy-affected
    ```

13. To test, open a web browser and issue any OData-specific query like this one:

    ```
    http://localhost:8180/odata4/<vdb>.<version>/<model>/<view>
    ```

    You will be redirected to perform SAML authentication.

Chapter 12. Patch Installation

12.1. 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 12.1. 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.

12.2. Patching Process

Red Hat JBoss Data Virtualization 6.3 and greater uses fuse-patch’s patch application system.

Previous versions of Red Hat JBoss Data Virtualization used other patch systems (Red Hat JBoss EAP’s for DV 6.2 and 6.1), and a manual system (for DV 6.0). Since it is part of the server, EAP’s system receives updates slowly, so for Red Hat JBoss Data Virtualizationa 6.3 it was necessary to use a separate patch system. Hence, there are now concurrent and separate EAP and DV patch streams.

To patch EAP, read the instructions found here: https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.4/html-single/Installation_Guide/#chap-Patching_and_Upgrading_JBoss_EAP_6

The self-executing jar patch distro-format can be executed in the same manner as the installer jar.

Red Hat JBoss Data Virtualization’s patch jars include the packages for all 6.3 versions up to the current version. This also enables a form of rollback, in that the package can be downgraded by re-installing the prior version.

Help text is provided by the jar when an argument is incorrect/unrecognised, or the --help switch is provided.

Procedure 12.2. Upgrading

1. Ensure the server is shut down.

2. To update to the latest version contained in the jar’s repository, run these commands:

```
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --
query-server
jboss-dv-6.3.0
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --
update jboss-dv
INFO  - Upgrade from jboss-dv-6.3.0 to jboss-dv-6.3.1
```

   Alternatively, you can manually specify the version using these commands:

```
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --
query-server
jboss-dv-6.3.0
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --
install jboss-dv-6.3.1
INFO  - Upgrade from jboss-dv-6.3.0 to jboss-dv-6.3.1
```

> **Note**
>
> The --server argument is optional. If it isn’t specified, the JBOSS_HOME environment variable, and then the current/working directory will be tried. A basic test will be undertaken to make sure the directory appears to be the server root, otherwise an error will be returned.

Procedure 12.3. Downgrading

1. Ensure the server is shut down.

2. Roll back by specifying a version lower than the current version as the install argument’s value:

```
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --
query-server
jboss-dv-6.3.1
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --
install jboss-dv-6.3.0
INFO  - Downgrade from jboss-dv-6.3.1 to jboss-dv-6.3.0
```

If you encounter an error during the upgrade process, you will see a message like this:

```
$ java -jar [path]/63-patching/jboss-dv-6.3.1-patch.jar --server dist/ --
update jboss-dv
INFO  - Upgrade from jboss-dv-6.3.0 to jboss-dv-6.3.1
ERROR - Attempt to override an already modified file
dataVirtualization/vdb/teiid-odata.war
ERROR - Attempt to override an already modified file
modules/system/layers/dv/org/jboss/teiid/api/main/teiid-api-8.12.5.redhat-
7.jar
ERROR - ERROR One or more checksum failures, aborting operation. Use --force
to override.
```

In this situation, you can revert the changes. To do so, run this command:

```
$ java -jar [path]/63-patching/jboss-dv-6.3.1-patch.jar --server dist/ --
update jboss-dv --force
INFO  - Upgrade from jboss-dv-6.3.0 to jboss-dv-6.3.1
WARN  - Overriding an already modified file: dataVirtualization/vdb/teiid-
odata.war
WARN  - Overriding an already modified file:
modules/system/layers/dv/org/jboss/teiid/api/main/teiid-api-8.12.5.redhat-
7.jar
```

Warning

An uninstall command is available, but Red Hat advises against using it. Running it will remove all files associated with the package name, not just that particular version. This is not a downgrade/rollback command (use install for that, as described above).

If you run this command accidentally, running "install" or "update" should put the files back into place.

Here are some miscellaneous commands:

To see the contents of the patch repository, run this command:

```
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --query-repository
jboss-dv-6.3.0
jboss-dv-6.3.1
```

This command lets you see the current packages on the server:

```
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --query-
server
jboss-dv-6.3.1
```

This shows you the paths owned by the currently installed package:

```
$ java -jar [path]/jboss-dv-6.3.1-patch.jar --server dist/ --query-server-
paths cli-scripts/
cli-scripts/ModeShape-domain.cli [jboss-dv-6.3.0]
cli-scripts/ModeShape-standalone.cli [jboss-dv-6.3.0]
cli-scripts/disable-welcome-root-domain.cli [jboss-dv-6.3.0]
cli-scripts/disable-welcome-root.cli [jboss-dv-6.3.0]
cli-scripts/teiid-add-database-logger-domain.cli [jboss-dv-6.3.0]
cli-scripts/teiid-add-database-logger.cli [jboss-dv-6.3.0]
cli-scripts/teiid-dashboard-add_datasource.cli [jboss-dv-6.3.0]
cli-scripts/teiid-dashboard-domain-add_datasource.cli [jboss-dv-6.3.0]
cli-scripts/teiid-deploy-dashboard-domain.cli [jboss-dv-6.3.0]
cli-scripts/teiid-deploy-dashboard.cli [jboss-dv-6.3.0]
cli-scripts/teiid-domain-auditcommand-logging.cli [jboss-dv-6.3.0]
cli-scripts/teiid-domain-mode-install.cli [jboss-dv-6.3.0]
cli-scripts/teiid-logger-ds.properties [jboss-dv-6.3.0]
cli-scripts/teiid-modeshape-domain.cli [jboss-dv-6.3.0]
cli-scripts/teiid-modeshape-standalone.cli [jboss-dv-6.3.0]
cli-scripts/teiid-standalone-auditcommand-logging.cli [jboss-dv-6.3.0]
cli-scripts/teiid-standalone-mode-install.cli [jboss-dv-6.3.0]
```

The argument value allows you to filter based on the the prefix of the paths. Passing an empty string (" or "" in most shells) will return all paths (unfiltered).

You can also view an audit log of previous actions using this command:

```
$ java -jar /path/to/jboss-dv-6.3.1-patch.jar --server some-server/ --audit-
log
# 13-Jul-2016 14:23:42
# Upgrade from jboss-dv-6.3.0 to jboss-dv-6.3.1
[properties]
[content]
ADD dataVirtualization/jdbc/modeshape-client-3.8.4.GA-redhat-8.jar
3027790150
DEL dataVirtualization/jdbc/modeshape-client-3.8.4.GA-redhat-9.jar 692252461
ADD dataVirtualization/jdbc/teiid-8.12.5.redhat-6-jdbc.jar 2751767369
DEL dataVirtualization/jdbc/teiid-8.12.5.redhat-7-jdbc.jar 181512329
ADD dataVirtualization/jdbc/teiid-hibernate-dialect-8.12.5.redhat-6.jar
2332531122
DEL dataVirtualization/jdbc/teiid-hibernate-dialect-8.12.5.redhat-7.jar
351948893
UPD dataVirtualization/logging/database-service.jar 2354483517
...

# 13-Jul-2016 14:25:18
# Downgrade from jboss-dv-6.3.1 to jboss-dv-6.3.0
[properties]
[content]
DEL dataVirtualization/jdbc/modeshape-client-3.8.4.GA-redhat-8.jar
3027790150
ADD dataVirtualization/jdbc/modeshape-client-3.8.4.GA-redhat-9.jar 692252461
DEL dataVirtualization/jdbc/teiid-8.12.5.redhat-6-jdbc.jar 2751767369
ADD dataVirtualization/jdbc/teiid-8.12.5.redhat-7-jdbc.jar 181512329
DEL dataVirtualization/jdbc/teiid-hibernate-dialect-8.12.5.redhat-6.jar
2332531122
ADD dataVirtualization/jdbc/teiid-hibernate-dialect-8.12.5.redhat-7.jar
351948893
UPD dataVirtualization/logging/database-service.jar 3678296797
...
```

12.3. 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 12.1. Severity Ratings of JBoss Security Patches

Severity	Description

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 12.1. 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 6.3.0-06	**Mon Sep 19 2016**	**David Le Sage**
Updates for 6.3.		
Revision 6.2.0-22	**Tue Aug 4 2015**	**David Le Sage**
Updates for 6.2.		