Securing IBM Operational Decision Manager

Contents

Securing Operational Decision Manager

IBM® Operational Decision Manager provides servers and components that are deployed in targeted IT environment for development, testing, and production, which integrate with other services. Knowing how to secure Operational Decision Manager applications and protect its data is critical to meet your company's compliance and security requirements.

Operational Decision Manager has three distribution options:

- Operational Decision Manager on premises
- Operational Decision Manager on IBM Cloud Private
- Operational Decision Manager on Cloud

Operational Decision Manager on premises can be deployed on your application server, such as WebSphere® Liberty. Note that you must design the application topology according to your needs and scalability. The servers and the connections between the applications must be configured and secured.

Operational Decision Manager on IBM Cloud Private provides a container-based deployment model. The Operational Decision Manager applications are Docker containers, and Kubernetes is used as the orchestration model. For more information, see Introducing ODM on IBM Cloud Private.

Operational Decision Manager on Cloud is either a separate cloud-based service or SaaS, which you can subscribe to. IBM provides this service per subscription. For more information, see IBM Operational Decision Manager on Cloud.

The following sections describe how you can secure Operational Decision Manager on your application server, although many sections are also applicable to Operational Decision Manager on IBM Cloud Private. All examples are provided by using WebSphere Liberty as an application server.

Security architecture

Identifying the components and the network connections among the components is the first step to understand the security architecture of Operational Decision Manager so that you know where you need to apply security controls.

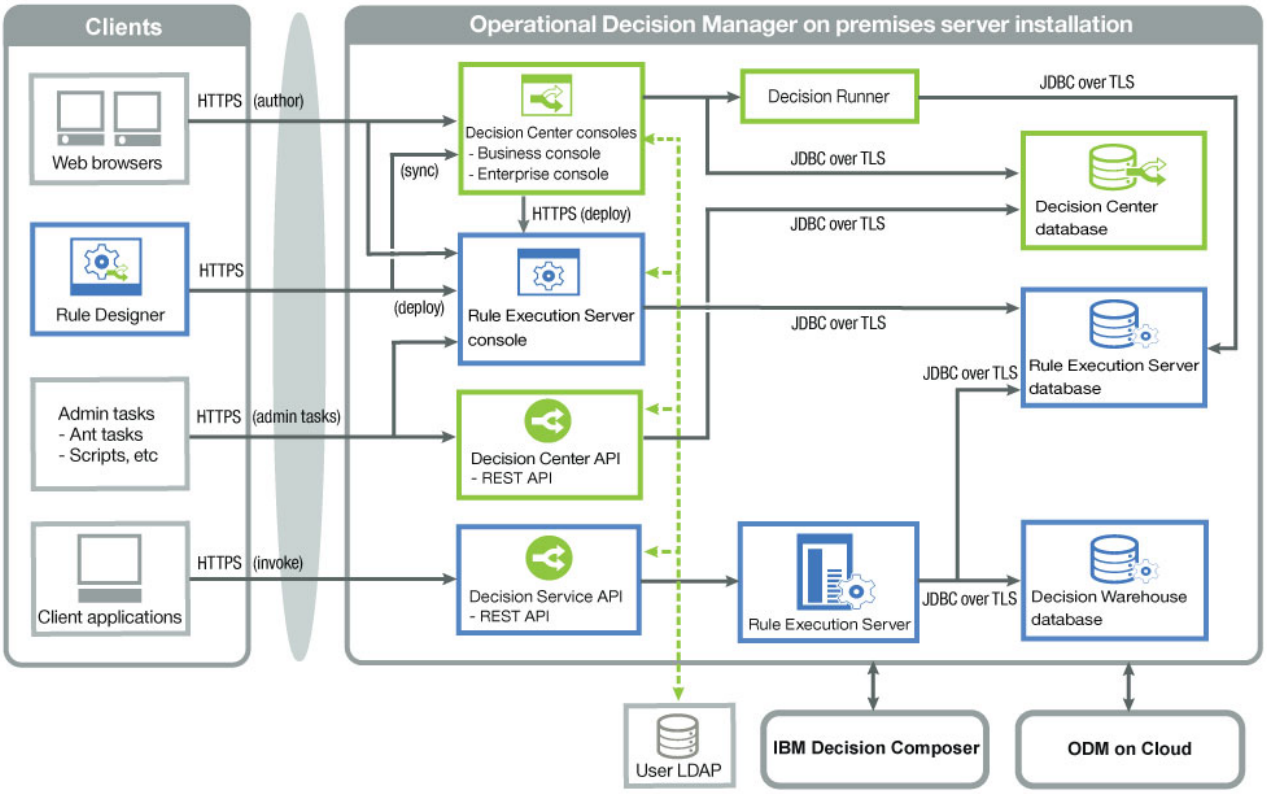

Figure 1. Operational Decision Manager security architecture

On the right side of Figure 1, you see the server side where Operational Decision Manager is installed on your application server.

- The following applications and APIs are publicly accessible:
  - Decision Center Business console
  - Decision Center Enterprise console
  - Rule Execution Server console
  - Decision Center API
  - Decision service API
- The following components are part of the installation, but they are not publicly accessible since they are backend components:
  - Decision model services

    The decision modeling feature is integrated into the Decision Center Business console in V8.10.0. Although there is a new WAR file because of this feature, there is no new user interface in the Business console.
  - Rule Execution Server
  - Decision Runner
- Data is stored in databases:
  - Decision Center database
  - Rule Execution Server database
  - Decision Warehouse database

  The connections between the applications and your data sources (databases and directory services) need to be secured by configuring Java™ database connectivity (JDBC) over TLS.

Operational Decision Manager is designed to use your company's directory service. You can bring your own LDAP directories for authentication and import users and groups in Decision Center, and assign permissions to groups.

On the left side of Figure 1, you see the client side. You can find the following client applications:

- Rule Designer interacts with the servers to synchronize projects and to deploy decision services. This component is provided by Operational Decision Manager.
- Web browsers are used to interact with three Operational Decision Manager web applications:
  - Decision Center Business console
  - Decision Center Enterprise console
  - Rule Execution Server console

  For information about supported web browsers, see the **Web Browsers** section in the **Prerequisites** tab in Operational Decision Manager Detailed System Requirements.
- Any command-line or client-side task to execute administrative tasks, such as ANT tasks, scripts, and cURL commands.
- Client applications invoke decision services at run time to execute decision services.

The following table summarizes which Operational Decision Manager component is the client and which is the server in the different network exchanges.

Table 1. Client/server exchanges

Client	Server	Purpose
Rule Designer	Decision Center	Synchronize rule projects.
Rule Designer	Rule Execution Server	Deploy decision services.
Decision Center (See Note below this table)	Rule Execution Server	Deploy decision services.
Web browser	• Decision Center Business console • Decision Center Enterprise console	Author and manage rules.
Web browser	Rule Execution Server console	Browse and deploy decision services.
Ant tasks	• Rule Execution Server API • Decision Center API	Any administrative tasks
Client applications	Decision service API	Call decision services.

Note: In the Decision Center and Rule Execution Server exchange, Decision Center is considered as a client since it connects to Rule Execution Server for deployment.

Configuring servers

After installing Operational Decision Manager, you configure the components on an application server to secure the communications with all potential clients.

Server certificates

Each server that runs a component of Operational Decision Manager must have a valid certificate.

You need to create a server certificate so that the Liberty server starts with the explicit server certificate assigned by you. The server certificate is contained in a keystore (`.jks`).

When the server starts with no explicit certificate, the Liberty server automatically produces a self-signed certificate with a domain assigned to your specific domain or local host. Avoid using the certificate automatically generated by the Liberty server as you do not control the certificate attributes.

When you create a certificate for a server, always check the following items:

- Certificate encryption attributes such as algorithm and length of the key
- Validity
- Server name, domain name, and common name
- Other attributes such as fingerprint

Creating server certificates

You must create an SSL certificate for your server.

Every server in your topology needs a keystore that contains a suitable server certificate, or a separate truststore that trusts all other servers.

You can use self-signed certificates or public certificates depending on your specific needs of your configuration:

- You can create an SSL keystore that contains a self-signed certificate by using the security utility tool, and then copy the keystore file (`.jks`) to the security folder of each server in your topology.
- You can also use the Java™ keystore tools to export the public certificate from the server, and import the certificate into the truststore of the client.

You can use the following tools to generate a server certificate in Java:

- Java **`keytool`** command
- WebSphere Liberty **`securityUtility`** command
- Graphic tools such as KeyStore Explorer

The following example shows you how to use the **`securityUtility createSSLCertificate`** command to create an SSL certificate for each server, specifying the server name and a password:

```
securityUtility createSSLCertificate --server=<server_name> --password=<your_password>
```

The command creates a `key.jks` keystore for the specified server as a result. For more information, see Creating SSL certificates from the command line in the WebSphere Liberty product documentation.

A certificate can be either self-signed or certificate authority (CA) signed, depending on your system configuration and the positioning of the server. That is, whether the server is public-facing or internal. The public-facing certificates are generally CA-signed, while the internal servers can rely on the self-signed certificates.

Self-signed certificate

If the certificate is self-signed, the browser prompts you to accept the insecure connection when you go to the URL for the first time. You need to answer `Yes` to see the URL and then view its certificate.

Client applications must be configured to accept a self-signed certificate from the server. For more information, see "Importing a server certificate to your environment" on page 8.

CA-signed certificate

When the certificate is CA-signed, the CA is already listed in the default certificate store of the browser or the client-side JVM. In this case, the connections to the server can be established without importing a certificate or having specific options.

Configuring server certificates

You must configure the SSL certificate in each server after creating it.

When you finish creating the SSL certificate, you confirm that the SSL feature is enabled in all of the `server.xml` files for the Liberty servers and add your password to the keystore definition:

```
<featureManager>
   <feature>ssl-1.0</feature>
</featureManager>
<keyStore id="defaultKeyStore" password="{xor}NjEsbjg3Kyw=" location="<myserver>.jks"/>
```

Add the *<myserver>*`.jks` file to the `/resources/security` folder in all the Liberty servers of your topology.

For more information, see Enabling SSL communication in Liberty in the WebSphere Liberty product documentation.

Configuring database connections

The databases that are used by the Operational Decision Manager components also need to be secure.

Consider the following aspects when you configure the databases:

- Secure data at rest: Have encrypted partitions or database-level encryption.
- Use TLS to connect to the databases: Use Java™ database connectivity (JDBC) over TLS.
- The databases must be highly available to avoid loss of data, especially for Decision Center.

Users, groups, and roles

Access control is an important part of securing your system. Configuring user access to the Operational Decision Manager applications involves defining users and groups of users, and then mapping them to the predefined roles.

Users

Users represent either people who work with Operational Decision Manager or system users who administer Operational Decision Manager. Users can be distinct

members or part of a group. Users or groups are authorized by associating them to roles.

Groups

Groups are collections of users. Administrators can assign all the users within the same group to a role, and by doing so, they do not have to manage users individually.

Roles

Different roles are assigned different levels of security access.

Decision Center roles

Decision Center has the following predefined roles:

Table 2. Decision Center user roles

Role	Description	Use
`rtsUser`	Standard user	Basic use.
`rtsInstaller`	Installer	Only required for the initial installation of Decision Center. Access the Installations Settings Wizard.
`rtsConfigManager`	Configuration manager	Has all the rights of the standard user, plus extra rights in the Business and Enterprise consoles. For example, create and edit deployment configurations.
`rtsAdministrator`	Administrator	Has all the rights of the standard and configuration manager users, plus extra rights in the Business and Enterprise consoles. For example, enforce security on decision services.

For more information, see Decision Center groups of users, Decision Center security, and "Decision Center authentication" on page 10.

Rule Execution Server roles

Rule Execution Server has the following predefined roles:

Table 3. Rule Execution Server user roles

Role	Description	Use
`resMonitors`	Monitor	Can view and monitor (read-only) decision services in the Rule Execution Server console.

Table 3. Rule Execution Server user roles (continued)

Role	Description	Use
`resDeployers`	Deployer	In addition to monitoring rights, can, for example, deploy decision services.
`resAdministrators`	Administrator	Full control in the Rule Execution Server console and on deployed resources.

For more information, see Rule Execution Server console and user roles. For information about how to bind groups of your user registry to the roles, see Deploying the Rule Execution Server management archive.

Default credentials

Operational Decision Manager is delivered with default credentials that are created for testing purposes. Default passwords should be changed immediately to strong passwords or removed.

Certain Ant tasks require user credentials to invoke tasks. See the following sections:

- Decision Center: Ant tasks
- Rule Execution Server: Ant tasks

Configuring clients

After installing Operational Decision Manager, you must configure the client side to secure the communications with the servers.

Enabling communication with self-signed certificates

You can use self-signed SSL certificates for internal or backend servers. You need to configure the **client** module in the client-server exchange if you want communication to work for self-signed certificates.

To enable communication to work with self-signed certificates, add the Java™ system property:

```
-Dcom.ibm.rules.httpclient.allowSelfSignedCertificates=true
```

The following table shows where to add the property depending on the client module:

Client module	Where
Rule Designer	In the `eclipse.ini` file of your Rule Designer installation as a single line at the end of the file.
Decision Center	In the configuration of the application server. For example, in WebSphere Liberty, in `jvm.options`.
Ant tasks	Set the Java system property in `ANT_OPTS`: `set ANT_OPTS=-Dcom.ibm.rules.httpclient.allowSelfSignedCertificates=true`

Note: The properties `-Dilog.rules.res.allowSelfSignedCertificate` and `-Dilog.rules.teamserver.allowSelfSignedCertificate` can still be used, but they are deprecated.

If you try to connect by using a certificate that is not trusted by the default JVM, an SSL handshake exception is thrown.

Importing a server certificate to your environment

After creating a keystore to with a security certificate in the `server.xml` file of each Liberty server, a client imports the certificate to its truststore to interact with the server.

Truststores are usually found in the `resources/security` folder of the Liberty server. In the simplest default configuration, the `key.jks` file acts as both the keystore and the truststore.

Another common configuration consists of one file as a keystore, and another as a truststore. You need to determine which file you need to modify.

Use of multiple truststores is possible. If you have this configuration, examine the server configuration to determine which files are to receive the certificate.

You can use the **`keytool`** command to import the certificate to the file.

Note: Before running the command, back up the key file in case any problems occur.

```
keytool -importcert \
     -file <certificate to trust> \
     -alias <alias for the certificate> \
     -keystore <name of the trustore> \
     -storepass <password for the truststore> \
     -storetype jks
```

After the import, you must restart the server.

For more information about how to obtain a public certificate from the other process and add the certificate to a Liberty truststore, see Adding trusted certificates in Liberty in the WebSphere Liberty product documentation.

Connecting Rule Designer with a server certificate

To be able to securely connect your Rule Designer to Decision Server and Decision Center, you need to establish a connection with a server certificate.

To connect to an Operational Decision Manager component with a server certificate, you use a `truststore.jks` file that integrates the server certificate.

1. Create a `truststore.jks` file, and place a copy of the file in your Rule Designer installation directory next to the `eclipse.ini` file.
2. Add the SSL properties at the end of your `eclipse.ini` file.

   Enter the following lines:

   ```
   -Djavax.net.ssl.trustStore=truststore.jks
   -Djavax.net.ssl.trustStorePassword=<truststore_password>
   ```

   Where *<truststore_password>* is the password that you define for your `truststore.jks` file.
3. Restart Rule Designer.

Test the connection between Rule Designer, the Rule Execution Server console and the Decision Center consoles to make sure that you are able to publish a decision service.

Another way to establish a connection is to import a server certificate file (`.cer`) into the JVM's `cacerts` file that is used to start Rule Designer. This method is not recommended because it changes Java configurations.

See the following example:

```
cd <JAVA_HOME>/jre/lib/security
keytool -importcert -file <certificate_file_name>.cer -keystore cacerts -storepass
 changeit -alias <your_server>
```

Where `changeit` is the fixed password that is used to protect the integrity of the `cacerts` file. You need to restart Rule Designer after the certificate is imported.

Verifying server certificates

When Operational Decision Manager is installed, you can go to an application by using a URL from a web browser. The first step to secure the communication is to examine the server certificate.

How to view the server certificate varies depending on your browser. In general, you can look at the server certificate by clicking the lock icon next to the URL in the browser.

When you view the server certificate, check the following information:

- Domain: the domain name must be the same as the one you intended to have.
- Validity period
- Encryption attributes

Host name verification

For added security, host name verification enforces a match between the certificate common name and the host name in the URL.

For development purposes, you can keep the default value of `com.ibm.rules.httpclient.verifyHostname=`**`false`** to disable host name verification.

In production, you must always enforce host name verification by changing the value of `com.ibm.rules.httpclient.verifyHostname` to `true`.

Note: When the value of `allowSelfSigned` is `true`, host name verification is disabled, regardless of whether you set `verifyHostname` to `true` or `false`.

Configuring proxy servers

You need to configure proxy servers if your enterprise uses an HTTP proxy to connect to Rule Execution Server or Decision Center. If you need to use an HTTP proxy server connection, you must configure the **client** module involved in the client-server exchange.

Configuring proxy servers in web browsers

You need to configure proxy servers in your web browsers if you use an HTTP proxy to connect to Rule Execution Server or Decision Center.

About this task

Specify the proxy server in the internet options of your operating system.

Note: See Java Networking and Proxies for a description of the different properties, which remain valid.

Procedure

For example, for Firefox:

1. Open Firefox.
2. Open **Menu** > **Preferences**.
3. In the **Network Settings** section, Click **Settings**.
4. Configure proxy access in the Connection Settings window.
5. Click **OK**.

Configuring proxy servers in Rule Designer

You need to configure proxy servers in Rule Designer if you use an HTTP proxy to connect to Rule Execution Server or Decision Center.

About this task

Add the proxy server to your Eclipse preferences.

Note: See Java Networking and Proxies for a description of the different properties, which remain valid.

Procedure

1. Click **Preferences** > **General** > **Network Connection**.
2. Set Active Provider to **Manual**.
3. Edit the HTTP and HTTPS proxy entries so that they correspond to your proxy server.
4. Save your changes.

Decision Center authentication and permission

Security features in Decision Center provides authentication to control access to user interface in the consoles. Decision Center also provides permission management to enable permissions on the different rule artifacts.

Decision Center authentication

When you sign in to Decision Center, you are associated with one of the predefined roles. Each role determines the parts of the user interface that is available in the Decision Center consoles. Different roles are assigned different levels of security, and the security access of users, or groups of users, can be controlled by assigning them to specific roles.

Granting access to Decision Center can be implemented by using the `ldapRegistry` feature in WebSphere Liberty to integrate your user registry in the LDAP directory.

A user registry stores user account information, such as user ID and password, and retrieves information about users and groups for security-related functions. LDAP directories are the most commonly used source of user and group data storage.

Note: Operational Decision Manager can access the user registry by using single sign-on (SSO). However, Security Assertion Markup Language (SAML) and OpenID Connect are not supported.

You can create as many groups as you need to organize your users in Decision Center. When you create a group, or import a group from the LDAP server, you place the users into groups, and then assign the groups to the roles. It is not possible to map a user to a role directly.

For more information about users, groups and roles, see Managing users from Decision Center.

For more information about how to configure user access to Decision Center, see Configuring user access to Decision Center.

For more information about configuration properties of `ldapRegistry`, see LDAP User Registry (ldapRegistry) in the WebSphere Liberty documentation.

Decision Center permission management

In Decision Center, permissions control how groups of users can access rule artifacts (action rules, decision tables).

In Decision Center, security defines which groups have access to the different branches of a decision service.

With authentication implemented, you specify, for each Decision Center group that can access the branch, what permissions they have to view, create, update, and delete which types of artifacts.

For more information about managing permissions in Decision Center, see Decision Center security and Permissions.

Optional server configurations

There are other configurations that you need to consider on the server side.

Configuring HTTP methods

Consider restrictions and configurations concerning security for certain HTTP methods.

Restricting the use of HTTP methods

Using certain HTTP methods can reveal the internal server configuration to outsiders, which makes your web applications more vulnerable. To restrict or forbid insecure or verbose HTTP methods such as OPTIONS and TRACE, you must make changes in the `web.xml` file of your web application.

You can specify a security constraint for HTTP methods in the `web.xml` file. See the following example that restricts two methods, OPTIONS and TRACE:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>restricted methods</web-resource-name>
        <url-pattern>/*</url-pattern>
        <http-method>OPTIONS</http-method>
        <http-method>TRACE</http-method>
    </web-resource-collection>
    <auth-constraint/>
</security-constraint>
```

Table 4. Description of the tags

Tag	Description
`<http-method>`	You specify an HTTP method that you want to restrict. You can specify one method in each set of tags.
`<auth-constraint/>`	This tag in the example indicates that no role can access the specified methods and these methods are forbidden.
`<url-pattern>`	You specify a URL pattern. If you specify a URL pattern instead of `/*`, the security constraint is applied to the URL pattern with specified HTTP methods.

You can change the setting of the `<auth-constraint>` tag and allow specific roles to access the restricted methods. See the following example:

```
<auth-constraint>
   <role-name>manager</role-name>
</auth-constraint>
```

The `manager` role has access to the methods here.

Overriding security restrictions of HTTP methods

When you use HTTP methods, consider the security aspects. For tighter security, some firewalls do not allow HTTP PUT or DELETE traffic.

To accommodate this restriction, you can send these requests in two ways:

- Use the **`X-Method-Override`** or **`X-HTTP-Method-Override`** HTTP header fields to channel a PUT or DELETE request through a POST request.
- Use the **`x-method-override`** or the **`x-http-method-override`** URI parameters.

  See the following example of the REST API for Rule Execution Server:

  ```
  POST /api/v1/ruleapps?...&x-method-override=PUT
  ```

Configuring the HTTP Strict Transport Security policy

A web application can be configured if it must be accessed only in HTTPS. In this case, any HTTP request can be redirected to HTTPS automatically. If it cannot be redirected to HTTPS, it generates an error.

You can configure the HTTP Strict Transport Security (HSTS) policy by using the following header:

```
Strict-Transport-Security: max-age=31536000; includeSubdomains; preload
```

In this example, the policy is set for one year (3600x24x365 seconds) with all of the subdomains included. When the policy is preloaded, it enables an application to redirect HTTP to HTTPS.

For more information about HSTS, see HTTP Strict Transport Security. You must choose the parameters for your web server carefully.

The Operational Decision Manager web applications do not provide an option to set this header. Therefore, the option must be managed at a higher level. You can enforce the HSTS policy by using one of the following methods:

- Use a reverse proxy server that sets the header.

- Use a preload list that redirects requests to HTTPS.
- Deactivate the HTTP port. In WebSphere Liberty, it can be done by using the following setting:

  ```
  <httpEndpoint id="defaultHttpEndpoint" httpPort="-1"/>
  ```

  Note: When the HTTP port is deactivated, an HTTP request generates an error.

Configuring cookies

All cookies that are created by the Operational Decision Manager applications contain the `HttpOnly` and `Secure` properties. These options must be set to mitigate the security risk and protect the cookies.

For more information about the `HttpOnly` and `Secure` options, see HttpOnly and SecureFlag in the Open Web Application Security Project (OWASP) documentation.

If you use WebSphere Liberty as your application server, check the following guidelines for cookies.

The Liberty server can use several different cookies. You must set all cookies to `HttpOnly` and `Secure` when you configure a Liberty application. The following cookies are the ones most often seen:

- `LtpaToken2` is used for Single Sign-On (SSO) among multiple Liberty servers.
- `JSESSIONID` contains an ID for the current session.

WebSphere Liberty also uses the following two cookies:

- `WASReqURL` contains the URL of the last visited HTTP request for the next SSO. It does not contain any sensitive information and can be handled like other cookies.
- `WASPostParam` contains the parameters of the last HTTP POST request. It does not contain any sensitive information and can be handled like other cookies.

Cookie `LtpaToken2`

You can use the `webAppSecurity` element in the `server.xml` file to configure web container application security for WebSphere Liberty:

```
<webAppSecurity
   ssoCookieName="LtpaToken2"
   ssoRequiresSSL="true"
   httpOnlyCookies="true"
   logoutOnHttpSessionExpire="true" ... />
```

Note: You must add `appSecurity-2.0`, `servlet-3.0`, and other required Liberty features to the `server.xml` file. For more information, see Application Security 2.0 in the WebSphere Liberty product documentation.

Property	Description
`ssoCookieName`	In general, you do not use `ssoCookieName` to change the cookie name. If you do so, the name must be replicated in all Liberty servers using the same SSO.

Property	Description
ssoRequiresSSL	Set this property to true so that the cookie LtpaToken2 is sent only over SSL to be secure. The default value is false.
httpOnlyCookies	When this property is set to true, it means that the cookie is set to HttpOnly. The default value is true.
logoutOnHttpSessionExpire	Set this property to true so that the users are logged out when the HTTP session timer expires. Set the path to / as you do not know the context root.

Cookie JSESSIONID

This cookie is used for configuring the HTTP session management. See the following example:

```
<httpSession
   cookieName="DCSESSIONID"
   cookieSecure="true"
   cookieHttpOnly="true"
   cookiePath="/"
... />
```

Property	Description
cookieName	The default value is DCSESSIONID.
cookieSecure	Set this property to true. The default value is false.
cookieHttpOnly	The default value is true. Do not change the default value.
cookiePath	The default value is /. If you know a specific cookie path, specify it here. Using useContextRootAsCookiePath might be fine. However, not all Operational Decision Manager applications might be in the same context root. A good practice can be to place all Operational Decision Manager applications under /odm/xxx, then use odm as the value of cookiePath.

Protecting decision services

Your client calls a decision service through the REST API or the SOAP API of Operational Decision Manager. By default, after you deploy a decision service to Rule Execution Server, anyone can invoke it without authentication or authorization. You can, however, secure the decision service with basic authentication so that anyone calling the decision service must first authenticate themselves.

Decision service URL patterns

Runtime URL patterns are used in Rule Execution Server.

- REST pattern: `https://<host>:<port>/<AppContextRoot>/DecisionService/rest/<rs_path>`
- SOAP pattern: `https://<host>:<port>/<AppContextRoot>/DecisionService/ws/<rs_path>`
- TEST pattern: `https://<host>:<port>/<AppContextRoot>/DecisionService/run.jsp?<TestParams>`

where

- `/rest/` represents the REST API
- `/ws/` represents the SOAP API
- `/run.jsp?` is used for testing ruleset in the Rule Execution Server console.

Protecting runtime decision services with basic authentication can be achieved based on these three URL patterns.

The ruleset testing feature is accessible through the Ruleset View page in the Rule Execution Server console:

1. Click **Retrieve HTDS Description File**.
2. Select **REST**.
3. Click **Test**.

Remember: The ruleset testing feature is not available for the SOAP API.

See the following examples when these URL patterns are developed:

- `https://app.example.com:9443/odm/test/DecisionService/rest/loans/1.0/miniloan/1.0`
- `https://app.example.com:9443/odm/test/DecisionService/ws/loans/1.0/miniloan/1.0`
- `https://app.example.com:9443/odm/test/DecisionService/run.jsp?path=/loans/1.0/miniloan/1.0&trace=false&type=WADL&kind=native`

The following example shows a complete URL format with a version:
`https://<host>:<port>/<AppContextRoot>/DecisionService/rest/v1/<rs_path>`

Table 1 explains each item in the URL format:

Table 5. Items in the URL patterns

Item	Name	Description
`<host>:<port>`	Hostname and port number	You can also configure the server to remove the port number (use a default port number). Example: `app.example.com:9443`

Table 5. Items in the URL patterns (continued)

Item	Name	Description
<AppContextRoot>	Application context root	You can specify one to provide a clearer path. It can be the product name, such as `odm`, followed by the environment name, such as `test`. Example: `odm/test`
`DecisionService`	Operational Decision Manager decision service	`DecisionService` Is always part of the URL.
`rest`, `ws`, or `run.jsp`	Fixed part to specify REST, SOAP, or testing ruleset URL	Followed by a ruleset path, appended in the path or provided as a parameter.
`v1`	API version	By default, the latest version of the API is launched (currently v1). The URLs invoke v2 when a new version v2 is available.
`<rs_path>`	Ruleset path	

Protecting the ruleset testing user interface

The TEST pattern is not a runtime API because the servlet (`run.jsp`) is accessible only when you use the interactive ruleset testing feature in the Rule Execution Server console. There is no default protection for the user interface. If you want to add security to the user interface, you must explicitly grant the access to user roles.

Before you begin

The ruleset testing feature is available only for the REST API. For information about the testing feature, see Testing a ruleset for REST execution

About this task

You can protect the TEST pattern by adding the `<security-constraint>` block in the `web.xml` deployment descriptor file. You must grant access to all three default roles (`resAdministrators`, `resMonitors`, and `resDeployers`) that are predefined in Rule Execution Server.

Procedure

See the following `<security-constraint>` block in the `web.xml` deployment descriptor file:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Decision Service Testing UI</web-resource-name>
        <url-pattern>/run.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>resAdministrators</role-name>
        <role-name>resMonitors</role-name>
        <role-name>resDeployers</role-name>
    </auth-constraint>
    <user-data-constraint>
```

```
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Default</realm-name>
</login-config>
```

Important: `CONFIDENTIAL` in the `transport-guarantee` tag means that the use of HTTP is required.
For more information about each subelement in the preceding example, see Specifying Security Constraints.
To invoke the decision service APIs with other user roles, you must add new roles.

Protecting decision services with basic authentication

The REST and SOAP patterns are used to invoke decision services. They can be protected by adding a new role that is dedicated to calling the APIs, and by configuring the `web.xml` file for the decision service `WAR` file for WebSphere Liberty (`DecisionService.war`).

Procedure

See the following example of `web.xml`:

```
<security-constraint>
    <web-resource-collection>
         <web-resource-name>Decision Service REST API</web-resource-name>
         <url-pattern>/rest/*</url-pattern>
         <url-pattern>/ws/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>resAdministrators</role-name>
        <role-name>resMonitors</role-name>
        <role-name>resDeployers</role-name>
        <role-name>resServiceUsers</role-name>
     </auth-constraint>
     <user-data-constraint>
          <transport-guarantee>CONFIDENTIAL</transport-guarantee>
     </user-data-constraint>
</security-constraint>
<login-config>
     <auth-method>BASIC</auth-method>
     <realm-name>Default</realm-name>
</login-config>
```

The example grants permissions to two URL patterns and four roles:

- Three predefined roles are required because the test user interface in the Rule Execution Server console invokes the REST (`/rest/`) and SOAP (`/ws/`) APIs internally:
  - `resAdministrators`
  - `resMonitors`
  - `resDeployers`
- A new role `resServiceUsers` dedicated for the API calls

It is a good idea to create the new role `resServiceUsers`. This new role is authorized to invoke decision services so that you can implement the basic authentication protection. To make it work, the incoming HTTP request must have a basic authentication header, and the user in the basic authentication must have the `resServiceUsers` role.
You can specify several `security-constraint` tags in a single `web.xml` file so that

you can protect different sets of URLs and use the same or different roles.
The `login-config` tag can be used only once in each `web.xml` file. In this tag, you specified `BASIC` for basic authentication:

```
<auth-method>BASIC</auth-method>
```

The following table shows possible values for `auth-method`:

auth-method value	**Description**
`BASIC`	For basic authentication.
`FORM`	For form-based authentication. This value is not suitable for API.
`CLIENT-CERT`	Client certificate
`DIGEST`	Digest authentication

Although it is possible to use `CLIENT-CERT` or `DIGEST` as an alternative way to protect the API, basic authentication is the simplest and the most convenient method of securing your decision services.

Creating one or several new roles

To create a new role `resServiceUsers` for decision services, you need to add a user to the basic registry.

Procedure

Add a user to the basic registry in the `server.xml` file. See the following example:

```
<basicRegistry id="basic" realm="customRealm">
    <user name="resAdmin" password="resAdmin" />
    <user name="resDeployer" password="resDeployer" />
    <user name="resMonitor" password="resMonitor" />
    <user name="resServiceUser" password="resServiceUser" />
    <group name="resAdministrators">
        <member name="resAdmin" />
    </group>
    <group name="resDeployers">
        <member name="resAdmin" />
        <member name="resDeployer" />
    </group>
    <group name="resMonitors">
        <member name="resAdmin" />
        <member name="resDeployer" />
        <member name="resMonitor" />
    </group>
    <group name="resServiceUsers">
        <member name="resServiceUser" />
    </group>
</basicRegistry>
<application type="war" id="DecisionService" name="DecisionService" location="${server.config.dir}/a
    <application-bnd>
        <security-role name="resAdministrators">
            <group name="resAdministrators" />
        </security-role>
        <security-role name="resDeployers">
            <group name="resDeployers" />
        </security-role>
        <security-role name="resServiceUsers">
            <group name="resServiceUsers" />
        </security-role>
     </application-bnd>
</application>
```

The example adds the new user `resServiceUser` with the password `resServiceUser`. The user is declared as a member of the `resServiceUsers` group.

Important: You must set a strong password. The password that is used earlier is only an example.
In the `application` block, you make an association between the `resServiceUsers` group and the `resServiceUser` role. The `DecisionService` application is now configured to use three roles, which are associated with the groups that have the same name.

Calling protected decision services

Before calling a protected decision service, you must have a user name and a password that can be used for the basic authentication header, and the right role that is authorized to call the corresponding decision services.

Using Java

The following Java code uses Apache HttpClient to set a basic authentication header:

```
HttpRequest request = new HttpPost("...");
String baHeader = username + ":" + password;
String base64 = Base64 .getEncoder().encodeToString(baHeader.getBytes());
request.setHeader("Authorization", "Basic " + base64);
```

Using cURL

Use `https://www.blitter.se/utils/basic-authentication-header-generator/` to generate a basic authentication header. Then, use a cURL command to make an invocation. See the following example:

```
#!/bin/bash

URL= https://myservice.ibmcloud.com/odm/dev
BASIC= dG90bzp0aXRp

echo Invoking: $URL/res/api/ruleapps?count=true
echo Response:
curl -k -v $URL/res/api/ruleapps?count=true \
-H "Authorization: Basic $BASIC"
echo
```

Optional protection configuration

You can implement other protection methods depending on your needs and environment.

Protecting specific RuleApps

You can configure security at the RuleApp level. However, it means that you have to update the `web.xml` deployment descriptor file every time a different RuleApp is deployed.

Before you begin

Restriction: If you frequently add or modify your RuleApps, you might not want to take this approach because the protection works only on that RuleApp.

About this task

Each deployed RuleApp leads to a different endpoint. For example, loan-related applications are in a RuleApp called `loans`, while pricing-related applications are in a RuleApp called `pricing`.

When they are deployed, these two RuleApps create different endpoints:

- `https://app.example.com:9443/odm/test/DecisionService/rest/loans/...`
- `https://app.example.com:9443/odm/test/DecisionService/rest/pricing/...`

The two distinct URL patterns `/rest/loans/*` and `/rest/pricing/*` can be used to protect the invocation of the corresponding RuleApps. They can also use different roles.

Procedure

See the following example of `web.xml`:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>REST API for Loan</web-resource-name>
        <url-pattern>/rest/loans/*</url-pattern>
        <url-pattern>/ws/loans/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>resLoansUsers</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>

<security-constraint>
    <web-resource-collection>
        <web-resource-name>REST API for Pricing</web-resource-name>
        <url-pattern>/rest/pricing/*</url-pattern>
        <url-pattern>/ws/pricing/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>resPricingUsers</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Specifying a per-environment application context root

You can specify the application context root per-environment basis. However, consider this advanced configuration only if you have environments that require the specific setup.

Procedure

The context root is specified in the `server.xml` file for all the `WAR` files that are run by the server, and it provides an extra path for the URL. See the following example:

```
<server>
    <application type="war" context-root="odm/test/DecisionService"location="${server.config.dir}/ap
        ...
    </application>
</server>
```

When you use this configuration in the `server.xml` file, the URL for a decision service looks like this:

```
https://app.example.com:9443/odm/test/DecisionService/rest/loans/1.0/miniloan/1.0
```

Then, the context root can be used to configure the `web.xml` file for protecting the HTDS on a per-environment basis by using a more specific URL:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Decision Service REST API</web-resource-name>
       <url-pattern>/odm/test/rest/*</url-pattern>
       <url-pattern>/odm/test/ws/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>resServiceTesters</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Security and Privacy by Design (SPbD)

Security and Privacy by Design (SPbD) at IBM is an agile set of focused security and privacy practices, including threat models, privacy assessments, security testing, and vulnerability management.

IBM developed a set of SPbD processes and tools that are used by all of its business units. For more information about the IBM Secure Engineering Framework (SEF) and SPbD, see the IBM Redbooks Security in Development - The IBM Secure Engineering Framework available in PDF format.

IBM also provides information about the features of Operational Decision Manager that you can configure, how to use the product securely, and what to consider to help your organization with GDPR readiness. For more information, see Deployment guidelines for GDPR readiness.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. 2016.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name, password or other personally identifiable information for purposes of session management,

authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also likely eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.