Red Hat JBoss Enterprise Application Platform 8-beta

Using single sign-on with JBoss EAP

Guide to using single sign-on to add authentication to applications deployed on JBoss EAP

Last Updated: 2022-12-16

Red Hat JBoss Enterprise Application Platform 8-beta Using single sign-on with JBoss EAP

Guide to using single sign-on to add authentication to applications deployed on JBoss EAP

Legal Notice

Abstract

Guide to using single sign-on to add authentication to applications deployed on JBoss EAP.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

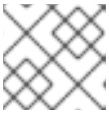
We appreciate your feedback on our documentation. To provide feedback, you can highlight the text in a document and add comments. Follow the steps in the procedure to learn about submitting feedback on Red Hat documentation.

Prerequisites

- Log in to the Red Hat Customer Portal.
- In the Red Hat Customer Portal, view the document in **Multi-page HTML** format.

Procedure

1. Click **Feedback** to see existing reader comments.



NOTE

The feedback feature is enabled only in the **Multi-page HTML** format.

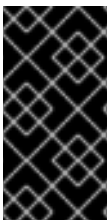
2. Highlight the section of the document where you want to provide feedback.
3. In the prompt menu that displays near the text you selected, click **Add Feedback**.
   A text box opens in the feedback section on the right side of the page.
4. Enter your feedback in the text box and click **Submit**.
   You have created a documentation issue.
5. To view the issue, click the issue tracker link in the feedback view.
6. Highlight the section of the document where you want to provide feedback.
7. In the prompt menu that displays near the text you selected, click **Add Feedback**.
   A text box opens in the feedback section on the right side of the page.
8. Enter your feedback in the text box and click **Submit**.
   You have created a documentation issue.
9. To view the issue, click the issue tracker link in the feedback view.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. OPENID CONNECT CONFIGURATION IN JBOSS EAP

When you secure your applications using an OpenID provider, you do not need to configure any security domain resources locally. The **elytron-oidc-client** subsystem provides a native OpenID Connect (OIDC) client in JBoss EAP to connect with OpenID providers. JBoss EAP automatically creates a virtual security domain for your application, based on your OpenID provider configurations.



IMPORTANT

It is recommended to use the OIDC client with Red Hat Single Sign-On. You can use other OpenID providers if they can be configured to use access tokens that are JSON Web Tokens (JWTs) and can be configured to use the RS256, RS384, RS512, ES256, ES384, or ES512 signature algorithm.

To enable the use of OIDC, you can configure either the **elytron-oidc-client** subsystem or an application itself. JBoss EAP activates the OIDC authentication as follows:

- When you deploy an application to JBoss EAP, the **elytron-oidc-client** subsystem scans the deployment to detect if the OIDC authentication mechanism is required.
- If the subsystem detects OIDC configuration for the deployment in either the **elytron-oidc-client** subsystem or the application deployment descriptor, JBoss EAP enables the OIDC authentication mechanism for the application.
- If the subsystem detects OIDC configuration in both places, the configuration in the **elytron-oidc-client** subsystem **secure-deployment** attribute takes precedence over the configuration in the application deployment descriptor.

Deployment configuration

To secure an application with OIDC by using a deployment descriptor, update the application's deployment configuration as follows:

- Create a file called **oidc.json** in the **WEB-INF** directory with the OIDC configuration information.

Example oidc.json contents

```
{
  "client-id" : "customer-portal", (1)
  "provider-url" : "http://localhost:8180/auth/realms/demo", (2)
  "ssl-required" : "external", (3)
   "credentials" : {
      "secret" : "234234-234234-234234" (4)
   }
}
```

(1) The name to identify the OIDC client with the OpenID provider.

(2) The OpenID provider URL.

(3) Require HTTPS for external requests.

4 The client secret that was registered with the OpenID provider.

- Set the **auth-method** property to **OIDC** in the application deployment descriptor **web.xml** file.

Example deployment descriptor update

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

Subsystem configuration

You can secure applications with OIDC by configuring the **elytron-oidc-client** subsystem in the following ways:

- Create a single configuration for multiple deployments if you use the same OpenID provider for each application.
- Create a different configuration for each deployment if you use different OpenID providers for different applications.

Example XML configuration for a single deployment:

```
<subsystem xmlns="urn:wildfly:elytron-oidc-client:1.0">
  <secure-deployment name="DEPLOYMENT_RUNTIME_NAME.war"> 1
    <client-id>customer-portal</client-id> 2
    <provider-url>http://localhost:8180/auth/realms/demo</provider-url> 3
    <ssl-required>external</ssl-required> 4
    <credential name="secret" secret="0aa31d98-e0aa-404c-b6e0-e771dba1e798" /> 5
  </secure-deployment
</subsystem>
```

1 The deployment runtime name.

2 The name to identify the OIDC client with the OpenID provider.

3 The OpenID provider URL.

4 Require HTTPS for external requests.

5 The client secret that was registered with the OpenID provider.

To secure multiple applications using the same OpenID provider, configure the **provider** separately, as shown in the example:

```
<subsystem xmlns="urn:wildfly:elytron-oidc-client:1.0">
  <provider name="${OpenID_provider_name}">
    <provider-url>http://localhost:8080/auth/realms/demo</provider-url>
    <ssl-required>external</ssl-required>
  </provider>
  <secure-deployment name="customer-portal.war"> 1
    <provider>${OpenID_provider_name}</provider>
    <client-id>customer-portal</client-id>
    <credential name="secret" secret="0aa31d98-e0aa-404c-b6e0-e771dba1e798" />
```

```
    </secure-deployment>
    <secure-deployment name="product-portal.war"> 2
        <provider>${OpenID_provider_name}</provider>
        <client-id>product-portal</client-id>
        <credential name="secret" secret="0aa31d98-e0aa-404c-b6e0-e771dba1e798" />
    </secure-deployment>
</subsystem>
```

1 A deployment: **customer-portal.war**

2 Another deployment: **product-portal.war**

Additional resources

- OpenID Connect specification
- **elytron-oidc-client** subsystem attributes
- OpenID Connect Libraries

CHAPTER 2. CONFIGURING RED HAT SINGLE SIGN-ON AS AN OPENID PROVIDER

Red Hat Single Sign-On is an identity and access management provider for securing web applications with single sign-on (SSO). It supports OpenID Connect (an extension to OAuth 2.0).

Prerequisites

- You have installed the Red Hat Single Sign-On server. For more information, see Installing the Red Hat Single Sign-On server in the Red Hat Single Sign-On *Getting Started Guide*.
- You have created a user in your Red Hat Single Sign-On server instance. For more information, see Creating a user in the Red Hat Single Sign-On *Getting Started Guide*.

Procedure

1. Start the Red Hat Single Sign-On server at a port other than 8080 because JBoss EAP default port is 8080.

   #### Syntax

   ```
   $ RH_SSO_HOME/bin/standalone.sh -Djboss.socket.binding.port-offset=<offset-number>
   ```

   #### Example

   ```
   $ /home/servers/rh-sso-7.4/bin/standalone.sh -Djboss.socket.binding.port-offset=100
   ```

2. Log in to the Admin Console at **http://localhost:<port>/auth/**. For example, **http://localhost:8180/auth/**.
3. To create a realm, in the Admin Console, hover over **Master**, and click **Add realm**.
4. Enter a name for the realm. For example, **example_realm**. Ensure that **Enabled** is **ON** and click **Create**.
5. Click **Users**, then click **Add user** to add a user to the realm.
6. Enter a user name. For example, **user1**. Ensure that User **Enabled** is **ON** and click **Save**.
7. Click **Credentials** to add a password to the user.
8. Set a password for the user. For example, **passwordUser1**. Toggle **Temporary** to **OFF** and click **Set Password**. In the confirmation prompt, click **Set password**.
9. Click **Clients**, then click **Create** to configure a client connection.
10. Enter a client ID. For example, **my_jbeap**. Ensure that **Client Protocol** is set to **openid-connect**, and click **Save**.
11. Click **Installation**, then select **Keycloak OIDC JSON** as the **Format Option** to see the connection parameters.

    ```
    {
      "realm": "example_realm",
    ```

```
  "auth-server-url": "http://localhost:8180/auth/",
  "ssl-required": "external",
  "resource": "my_jbeap",
  "public-client": true,
  "confidential-port": 0
}
```

When configuring your JBoss EAP application to use Red Hat Single Sign-On as the identity provider, you use the parameters as follows:

```
"provider-url" : "http://localhost:8180/auth/realms/example_realm",
"ssl-required": "external",
"client-id": "my_jbeap",
"public-client": true,
"confidential-port": 0
```

12. Click **Clients**, click **Edit** next to **my_jbeap** to edit the client settings.

13. In **Valid Redirect URIs**, enter the URL where the page should redirect after authentication is successful.
For this example, set this value to **http://localhost:8080/simple-webapp-example/secured/*** and then click **Save**.

Additional resources

- Creating a realm and a user

CHAPTER 3. CREATING A MAVEN PROJECT FOR WEB-APPLICATION DEVELOPMENT

For creating a web-application, create a Maven project with the required dependencies and the directory structure.

Prerequisites

- You have installed Maven. For more information, see Downloading Apache Maven.

Procedure

1. Set up a Maven project using the **mvn** command. The command creates the directory structure for the project and the **pom.xml** configuration file.

   #### Syntax

   ```
   $ mvn archetype:generate \
   -DgroupId=${group-to-which-your-application-belongs} \
   -DartifactId=${name-of-your-application} \
   -DarchetypeGroupId=org.apache.maven.archetypes \
   -DarchetypeArtifactId=maven-archetype-webapp \
   -DinteractiveMode=false
   ```

   #### Example

   ```
   $ mvn archetype:generate \
   -DgroupId=com.example.app \
   -DartifactId=simple-webapp-example \
   -DarchetypeGroupId=org.apache.maven.archetypes \
   -DarchetypeArtifactId=maven-archetype-webapp \
   -DinteractiveMode=false
   ```

2. Navigate to the application root directory:

   #### Syntax

   ```
   $ cd <name-of-your-application>
   ```

   #### Example

   ```
   $ cd simple-webapp-example
   ```

3. Replace the content of the generated **pom.xml** file with the following text:

   ```
   <?xml version="1.0" encoding="UTF-8"?>

   <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
     <modelVersion>4.0.0</modelVersion>
   ```

```
  <groupId>com.example.app</groupId>
  <artifactId>simple-webapp-example</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>simple-webapp-example Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>jakarta.servlet</groupId>
      <artifactId>jakarta.servlet-api</artifactId>
      <version>6.0.0</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.wildfly.security</groupId>
      <artifactId>wildfly-elytron-auth-server</artifactId>
      <version>1.19.0.Final</version>
    </dependency>
  </dependencies>

  <build>
    <finalName>${project.artifactId}</finalName>
    <plugins>
      <plugin>
        <groupId>org.wildfly.plugins</groupId>
        <artifactId>wildfly-maven-plugin</artifactId>
        <version>2.1.0.Final</version>
      </plugin>
    </plugins>
  </build>

</project>
```

Verification

- In the application root directory, enter the following command:

```
$ mvn install
```

You get an output similar to the following:

```
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

```
[INFO] Total time: 0.795 s
[INFO] Finished at: 2022-04-28T17:39:48+05:30
[INFO] ------------------------------------------------------------------------
```

You can now create a web-application.

CHAPTER 4. CREATING A WEB APPLICATION

Create a web application containing a servlet that returns the user name obtained from the logged-in user's principal and attributes. If there is no logged-in user, the servlet returns the text "NO AUTHENTICATED USER".

Prerequisites

- You have created a Maven project.
- JBoss EAP is running.

Procedure

1. Create a directory to store the Java files.

   #### Syntax

   ```
   $ mkdir -p src/main/java/<path_based_on_artifactID>
   ```

   #### Example

   ```
   $ mkdir -p src/main/java/com/example/app
   ```

2. Navigate to the new directory.

   #### Syntax

   ```
   $ cd src/main/java/<path_based_on_artifactID>
   ```

   #### Example

   ```
   $ cd src/main/java/com/example/app
   ```

3. Create a file **SecuredServlet.java** with the following content:

   ```
   package com.example.app;

   import java.io.IOException;
   import java.io.PrintWriter;
   import java.security.Principal;
   import java.util.ArrayList;
   import java.util.Collection;
   import java.util.Iterator;
   import java.util.List;
   import java.util.Set;

   import jakarta.servlet.ServletException;
   import jakarta.servlet.annotation.WebServlet;
   import jakarta.servlet.http.HttpServlet;
   import jakarta.servlet.http.HttpServletRequest;
   import jakarta.servlet.http.HttpServletResponse;
   import org.wildfly.security.auth.server.SecurityDomain;
   ```

```
import org.wildfly.security.auth.server.SecurityIdentity;
import org.wildfly.security.authz.Attributes;
import org.wildfly.security.authz.Attributes.Entry;
/**
 * A simple secured HTTP servlet. It returns the user name and
 * attributes obtained from the logged-in user's Principal. If
 * there is no logged-in user, it returns the text
 * "NO AUTHENTICATED USER".
 */

@WebServlet("/secured")
public class SecuredServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        try (PrintWriter writer = resp.getWriter()) {

         Principal user = req.getUserPrincipal();
         SecurityIdentity identity = SecurityDomain.getCurrent().getCurrentSecurityIdentity();
         Attributes identityAttributes = identity.getAttributes();
         Set <String> keys = identityAttributes.keySet();
         String attributes = "<ul>";

         for (String attr : keys) {
          attributes += "<li> " +  attr + " : " + identityAttributes.get(attr).toString() + "</li>";
         }

         attributes+="</ul>";
         writer.println("<html>");
         writer.println("  <head><title>Secured Servlet</title></head>");
         writer.println("  <body>");
         writer.println("    <h1>Secured Servlet</h1>");
         writer.println("    <p>");
         writer.print(" Current Principal '");
         writer.print(user != null ? user.getName() : "NO AUTHENTICATED USER");
         writer.print("'");
         writer.print(user != null ? "\n" + attributes : "");
         writer.println("    </p>");
         writer.println("  </body>");
         writer.println("</html>");
        }
    }

}
```

4. In the application root directory, compile your application with the following command:

```
$ mvn package
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.015 s
[INFO] Finished at: 2022-04-28T17:48:53+05:30
[INFO] ------------------------------------------------------------------------
```

5. Deploy the application.

```
$ mvn wildfly:deploy
```

Verification

- In a browser, navigate to **http://localhost:8080/simple-webapp-example/secured**.
  You get the following message:

```
Secured Servlet
Current Principal 'NO AUTHENTICATED USER'
```

  Because no authentication mechanism is added, you can access the application.

You can now secure this application by using a security domain so that only authenticated users can access it.

CHAPTER 5. SECURING A WEB APPLICATION USING OPENID CONNECT

You can secure an application by either updating its deployment configuration or by configuring the **elytron-oidc-client subsystem**.

If you use the application created in the procedure, Creating a web application , the value of the Principal comes from the ID token from the OpenID provider. By default, the Principal is the value of the "sub" claim from the token. You can specify which claim value from the ID token to use as the Principal in one of the following:

- The **elytron-oidc-client** subsystem attribute **principal-attribute**.
- **The oidc.json file**.

Prerequisites

- You have deployed applications on JBoss EAP.

Procedure

1. Configure the application's **web.xml** to protect the application resources.

   #### Syntax

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>

 <!-- Define the security constraints for the application resources.
     Specify the URL pattern for which a challenge is -->

 <security-constraint>
      <web-resource-collection>
         <web-resource-name><!-- Name of the resources to protect --></web-resource-name>
         <url-pattern> <!-- The URL to protect  --></url-pattern>
      </web-resource-collection>

      <!-- Define the role that can access the protected resource -->
      <auth-constraint>
         <role-name> <!-- Role name as defined in the security domain --></role-name>
         <!-- To disable authentication you can use the wildcard *
           To authenticate but allow any role, use the wildcard **. -->
      </auth-constraint>
   </security-constraint>

   <login-config>
      <auth-method>
       <!-- The authentication method to use. Can be:
        BASIC
        CLIENT-CERT
        DIGEST
        FORM
```

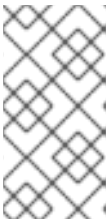
```
        SPNEGO
        -->
      </auth-method>

      <realm-name><!-- The name of realm to send in the challenge  --></realm-name>
    </login-config>
  </web-app>
```

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    metadata-complete="false">

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>secured</web-resource-name>
            <url-pattern>/secured</url-pattern>
        </web-resource-collection>

        <auth-constraint>
            <role-name>Admin</role-name>
        </auth-constraint>
    </security-constraint>

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

    <security-role>
        <role-name>*</role-name>
    </security-role>
</web-app>
```

In this example, only the users with the role **Admin** can access the application.

2. To secure the application with OpenID Connect, either update the deployment configuration or configure the **elytron-oidc-client** subsystem.



NOTE

If you configure OpenID Connect in both the deployment configuration and the **elytron-oidc-client** subsystem, the configuration in the **elytron-oidc-client** subsystem **secure-deployment** attribute takes precedence over the configuration in the application deployment descriptor.

- Updating the deployment configuration:

  i. Create a file **oidc.json** in the **WEB-INF** directory, like this:

```
{
```

```
  "provider-url" : "http://localhost:8180/auth/realms/example_realm",
  "ssl-required": "external",
  "client-id": "my_jbeap",
  "public-client": true,
  "confidential-port": 0
}
```

ii. Update the deployment descriptor **web.xml** file with the following text to declare that this application uses OIDC:

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

- Configuring the **elytron-oidc-client** subsystem:
  - To secure your application, use the following management CLI command:

```
/subsystem=elytron-oidc-client/secure-deployment=simple-oidc-example.war/:add(client-id=my_jbeap,provider-url=http://localhost:8180/auth/realms/example_realm,public-client=true,ssl-required=external)
```

3. In the application root directory, compile your application with the following command:

```
$ mvn package
```

4. Deploy the application.

```
$ mvn wildfly:deploy
```

Verification

1. In a browser, navigate to **http://localhost:8080/simple-webapp-example/secured**. You are redirected to Red Hat Single Sign-On login page.
2. Log in with your credentials. For example:

```
username: user1
password: passwordUser1
```

You get the following output:

```
Forbidden
```

The redirection to Red Hat Single Sign-On login page confirms that the OIDC connection succeeds and the output confirms that users without the role **Admin** cannot access the application. To add the role **Admin** to the user **user1**, see Creating and assigning user roles in Red Hat Single Sign-On .

Additional resources

- **elytron-oidc-client** subsystem attributes

CHAPTER 6. CREATING AND ASSIGNING USER ROLES IN RED HAT SINGLE SIGN-ON

Red Hat Single Sign-On is an identity and access management provider for securing your web applications with single sign-on (SSO). You can define users and assign roles in Red Hat Single Sign-On.

Prerequisites

- You have secured your application using OpenID Connect with Red Hat Single Sign-On as the identity provider.

Procedure

1. Log in to the admin console at **http://localhost:<port>/auth/**. For example, **http://localhost:8180/auth/**.
2. Click the realm you use to connect with JBoss EAP. For example, *example_realm*.
3. Click **Clients**, then click the **client-name** you configured for JBoss EAP. For example, *my_jbeap*.
4. Click **Roles**, then **Add Role**.
5. Enter a role name, such as *Admin*, then click **Save**. This is the role name you configure in JBoss EAP for authorization.
6. Click **Users**, then **View all users**.
7. Click an ID to assign the role you created. For example, click the ID for *user1*.
8. Click **Role Mappings**. In the **Client Roles** field, select the **client-name** you configured for JBoss EAP. For example, *my_jbeap*.
9. In **Available Roles**, select a role to assign. For example, *admin*. Click **Add selected**.

Verification

1. If your application is already deployed, undeploy the aplication and deploy it again. In the application root directory, enter the following commands:

```
$ mvn wildfy:undeploy
$ mvn wildfy:deploy
```

2. In a browser, navigate to the application URL. For example, **http://localhost:8080/simple-webapp-example/secured**.
   You are redirected to Red Hat Single Sign-On login page.
3. Log in with your credentials. For example:

```
username: user1
password: passwordUser1
```

   You get the following output:

```
Secured Servlet
```

```
Current Principal 'cc02dfd3-198d-47e4-a9a9-021c5492e230'

Roles : [offline_access, default-roles-example_realm, uma_authorization, Admin]
```

The value of the Principal comes from the ID token from the OpenID provider. The Principal here is the value of the "sub" claim from the token.

Users with the required role can log in to your application.

Additional resources

- Assigning permissions and access using roles and groups in Red Hat Single Sign-On

CHAPTER 7. ELYTRON-OIDC-CLIENT SUBSYSTEM ATTRIBUTES

The **elytron-oidc-client** subsystem provides attributes to configure its behavior.

Table 7.1. elytron-oidc-client subsystem attributes

Attribute	Description
provider	Configuration for an OpenID Connect provider.
secure-deployment	A deployment secured by an OpenID Connect provider.
realm	Configuration for a Red Hat Single Sign-On realm. This is provided for convenience. You can copy the configuration in the keycloak client adapter and use it here. Using the **provider** is recommended instead.

Use the three **elytron-oidc-client** attributes for the following purposes:

- **provider**: For configuring the OpenID Connect provider. For more information, see **provider** attributes.
- **secure-deployment**: For configuring the deployment secured by an OpenID Connect. For more information, see **secure-deployment** attributes
- **realm**: For configuring Red Hat Single Sign-On. For more information, see **realm** attributes. The use of **realm** is not recommended. It is provided for convenience. You can copy the configuration in the keycloak client adapter and use it here. Using the **provider** attribute is recommended instead.

Table 7.2. provider attributes

Attribute	Default value	Description
allow-any-hostname	**false**	If you set the value to **true**, hostname verification is skipped when communicating with the OpenID provider. This is useful when testing. Do not set this to **ture** in a production environment.
always-refresh-token		If set to **true**, JBoss EAP refreshes tokens on every web request.
auth-server-url		The base URL of the Red Hat Single Sign-On realm authorization server. If you use this attribute, you must also define the **realm** attribute. You can alternatively use the **provider-url** attribute to provide both base URL and the realm in a single attribute.

Attribute	Default value	Description
autodetect-bearer-only	**false**	Set whether to automatically detect bearer-only requests. When a bearer-only request is received and **autodetect-bearer-only** is set to **true**, the application cannot participate in browser logins.
client-id		The client-id of JBoss EAP registered with the OpenID provider.
client-key-password		If you specify **client-keystore**, specify it's password in this attribute.
client-keystore		If your application communicates with the OpenID provider over HTTPS, set the path to the client keystore in this attribute.
client-keystore-password		If you specify the **client keystore**, provide the password for accessing it in this attribute.
confidential-port	**8443**	Specify the confidential port (SSL/TLS) used by the OpenID provider.
connection-pool-size		Specify the connection pool size to be used when communicating with the OpenID provider.
connection-timeout-millis		Specify the timeout for establishing a connection with the remote host in milliseconds. The minimum is **-1L**, and the maximum **2147483647L**.**-1L** indicates that the value is undefined, which is the default.
connection-ttl-millis		Specify the amount of time in milliseconds for the connection to be kept alive. The minimum is **-1L**, and the maximum **2147483647L**. **-1L** indicates that the value is undefined, which is the default.
cors-allowed-headers		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the **Access-Control-Allow-Headers** header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-allowed-methods		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the Access-Control-Allow-Methods header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-exposed-headers		If CORS is enabled, this sets the value of the Access-Control-Expose-Headers header. This should be a comma-separated string. This is optinal. If not set, this header is not returned in CORS responses.

Attribute	Default value	Description
cors-max-age		Set the value for Cross-Origin Resource Sharing (CORS) Max-Age header. The value can be between **-1L** and **2147483647L**. This attribute only takes effect if **enable-cors** is set to **true**.
disable-trust-manager		Specify whether or not to make use of a trust manager when communicating with the OpenID provider over HTTPS.
enable-cors	**false**	Enable Red Hat Single Sign-On Cross-Origin Resource Sharing (CORS) support.
expose-token	**false**	If set to **true**, an authenticated browser client can obtain the signed access token, through a Javascript HTTP invocation, via the URL **root/k_query_bearer_token**. This is optional. This is specific to Red Hat Single Sign-On.
ignore-oauth-query-parameter	**false**	Disable query parameter parsing for access_token.
principal-attribute		Specify which claim value from the ID token to use as the principal for the identity
provider-url		Specify the OpenID provider URL.
proxy-url		Specify the URL for the HTTP proxy if you use one.
realm-public-key		Specify the public key of the realm.
register-node-at-startup	**false**	If set to **true**, a registration request is sent to Red Hat Single Sign-On. This attribute is useful only when your application is clustered.
register-node-period		Specify how often to re-register the node.
socket-timeout-millis		Specify the timeout for socket waiting for data in milliseconds.
ssl-required	**external**	Specify whether communication with the OpenID provider should be over HTTPS. The value can be one of the following: ● **all** - all communication happens over HTTPS. ● **external** - Only the communication with external clients happens over HTTPs. ● **none** - HTTPs is not used.

Attribute	Default value	Description
token-signature-algorithm	**RS256**	Specify the token signature algorithm used by the OpenID provider. The supported algorithms are: • **RS256** • **RS384** • **RS512** • **ES256** • **ES384** • **ES512**
token-store		Specify cookie or session storage for auth-session data.
truststore		Specify the truststore used for client HTTPS requests.
truststore-password		Specify the truststore password.
verify-token-audience	**false**	If set to **true**, then during bearer-only authentication,verify if token contains this client name (**resource**) as an audience.

Table 7.3. **secure-deployment** attributes

Attribute	Default value	Description
allow-any-hostname	**false**	If you set the value to **true**, hostname verification is skipped when communicating with the OpenID provider. This is useful when testing. Do not set this to **ture** in a production environment.
always-refresh-token		If set to **true**, JBoss EAP refreshes tokens on every web request.
auth-server-url		The base URL of the Red Hat Single Sign-On realm authorization server You can alternatively use the **provider-url** attribute.

Attribute	Default value	Description
autodetect-bearer-only	**false**	Set whether to automatically detect bearer-only requests. When a bearer-only request is received and **autodetect-bearer-only** is set to **true**, the application cannot participate in browser logins.
bearer-only	false	Set this to **true** to secure the application with Bearer Token authentication. When Bearer Token authentication is enabled, users are not redirected to the OpenID provider to log in; instead, the **elytron-oidc-client** subsystem attempts to verify the user's bearer token. The default value for **bearer-only** is **false**.
client-id		The client-id of JBoss EAP registered with the OpenID provider.
client-key-password		If you specify **client-keystore**, specify it's password in this attribute.
client-keystore		If your application communicates with the OpenID provider over HTTPS, set the path to the client keystore in this attribute.
client-keystore-password		If you specify the **client keystore**, provide the password for accessing it in this attribute.
confidential-port	**8443**	Specify the confidential port (SSL/TLS) used by OpenID provider.
connection-pool-size		Specify the connection pool size to be used when communicating with the OpenID provider.

Attribute	Default value	Description
connection-timeout-millis		Specify the timeout for establishing a connection with the remote host in milliseconds. The minimum is **-1L**, and the maximum **2147483647L**. **-1L** indicates that the value is undefined, which is the default.
connection-ttl-millis		Specify the amount of time in milliseconds for the connection to be kept alive. The minimum is **-1L**, and the maximum **2147483647L**. -1L indicates that the value is undefined, which is the default.
cors-allowed-headers		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the **Access-Control-Allow-Headers** header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-allowed-methods		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the **Access-Control-Allow-Methods** header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-exposed-headers		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the **Access-Control-Expose-Headers** header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-max-age		Set the value for Cross-Origin Resource Sharing (CORS) Max-Age header. The value can be between **-1L** and **2147483647L**. This attribute only takes effect if `enable-

Attribute	Default value	Description
credential		Specify the credential to use to communicate with the OpenID provider.
disable-trust-manager		Specify whether or not to make use of a trust manager when communicating with the OpenID provider over HTTPS.
enable-cors	**false**	Enable Red Hat Single Sign-On Cross-Origin Resource Sharing (CORS) support.
enable-basic-auth	**false**	Enable Basic Authentication to specify the credentials to be used to obtain a bearer token.
expose-token	**false**	If set to **true**, an authenticated browser client can obtain the signed access token, through a Javascript HTTP invocation, via the URL **root/k_query_bearer_token**. This is optional.This is specific to Red Hat Single Sign-On.
ignore-oauth-query-parameter	**false**	Disable query parameter parsing for access_token.
min-time-between-jwks-requests		If adapter recognizes a token signed by an unknown public key, JBoss EAP tries to download new public key from the **elytron-oidc-client** server. However, JBoss EAP deosn't try to download new public key if it has already tried it in less than the value, in seconds, that you set for this attribute. The value can be between **-1L** and **2147483647L**.
principal-attribute		Specify which claim value from the ID token to use as the principal for the identity
provider		Specify the OpenID provider.
provider-url		Specify the OpenID provider URL.

Attribute	Default value	Description
proxy-url		Specify the URL for the HTTP proxy if you use one.
public-client	**false**	If set to **true**, no client credentials are sent when communicating with the OpenID provider. This is optional.
realm		The realm with which to connect in Red Hat Single Sign-On.
realm-public-key		Specify the public key of the realm.
redirect-rewrite-rule		Specify the rewrite rule to apply to the redirect URI.
register-node-at-startup	**false**	If set to **true**, a registration request is sent to Red Hat Single Sign-On. This attribute is useful only when your application is clustered.
register-node-period		Specify how often to re-register the node.
resource		Specify the name of the application you are securing with OIDC. Alternatively, you can specify the **client-id**.
socket-timeout-millis		Specify the timeout for socket waiting for data in milliseconds.
ssl-required	**external**	Specify whether communication with the OpenID provider should be over HTTPS. The value can be one of the following: • **all** - all communication happens over HTTPS. • **external** - Only the communication with external clients happens over HTTPs. • **none** - HTTPs is not used.

Attribute	Default value	Description
token-minimum-time-to-live		The adapter refreshes the token if the current token is expired or is to expire within the amount of time you set in seconds.
token-signature-algorithm	**RS256**	Specify the token signature algorithm used by the OpenID provider. The supported algorithms are: • **RS256** • **RS384** • **RS512** • **ES256** • **ES384** • **ES512**
token-store		Specify cookie or session storage for auth-session data.
truststore		Specify the truststore used for adapter client HTTPS requests.
truststore-password		Specify the truststore password.
turn-off-change-session-id-on-login	**false**	The session id is changed by default on a successful login. Set the value to **true** to turn this off.
use-resource-role-mappings	**false**	Use resource-level permissions obtained from token.
verify-token-audience	**false**	If set to **true**, then during bearer-only authentication, the adapter verifies if token contains this client name (**resource**) as an audience.

Table 7.4. **realm** attributes

Attribute	Default value	Description

Attribute	Default value	Description
allow-any-hostname	**false**	If you set the value to **true**, hostname verification is skipped when communicating with the OpenID provider. This is useful when testing. Do not set this to **ture** in a production environment.
always-refresh-token		If set to **true**, JBoss EAP refreshes tokens on every web request.
auth-server-url		The base URL of the Red Hat Single Sign-On realm authorization server You can alternatively use the **provider-url** attribute.
autodetect-bearer-only	**false**	Set whether to automatically detect bearer-only requests. When a bearer-only request is received and **autodetect-bearer-only** is set to **true**, the application cannot participate in browser logins.
client-key-password		If you specify **client-keystore**, specify it's password in this attribute.
client-keystore		If your application communicates with the OpenID provider over HTTPS, set the path to the client keystore in this attribute.
client-keystore-password		If you specify the **client keystore**, provide the password for accessing it in this attribute.
confidential-port	**8443**	Specify the confidential port (SSL/TLS) used by Red Hat Single Sign-On.
connection-pool-size		Specify the connection pool size to be used when communicating with Red Hat Single Sign-On.

Attribute	Default value	Description
connection-timeout-millis		Specify the timeout for establishing a connection with the remote host in milliseconds. The minimum is **-1L**, and the maximum **2147483647L**. **-1L** indicates that the value is undefined, which is the default.
connection-ttl-millis		Specify the amount of time in milliseconds for the connection to be kept alive. The minimum is **-1L**, and the maximum **2147483647L**. **-1L** indicates that the value is undefined, which is the default.
cors-allowed-headers		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the Access-Control-Allow-Headers header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-allowed-methods		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the **Access-Control-Allow-Methods header**. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-exposed-headers		If Cross-Origin Resource Sharing (CORS) is enabled, this sets the value of the **Access-Control-Expose-Headers** header. This should be a comma-separated string. This is optional. If not set, this header is not returned in CORS responses.
cors-max-age		Set the value for Cross-Origin Resource Sharing (CORS) Max-Age header. The value can be between **-1L** and **2147483647L**. This attribute only takes effect if **enable-cors** is set to **true**.

Attribute	Default value	Description
disable-trust-manager		Specify whether or not to make use of a trust manager when communicating with the OpenID provider over HTTPS._
enable-cors	**false**	Enable {RHProductShortName} Cross-Origin Resource Sharing (CORS) support.
expose-token	**false**	If set to **true**, an authenticated browser client can obtain the signed access token, through a Javascript HTTP invocation, via the URL **root/k_query_bearer_token**. This is optional.
ignore-oauth-query-parameter	**false**	Disable query parameter parsing for access_token.
principal-attribute		Specify which claim value from the ID token to use as the principal for the identity
provider-url		Specify the OpenID provider URL.
proxy-url		Specify the URL for the HTTP proxy if you use one.
realm-public-key		Specify the public key of the realm.
register-node-at-startup	**false**	If set to **true**, a registration request is sent to Red Hat Single Sign-On. This attribute is useful only when your application is clustered.
register-node-period		Specify how often to re-register the node.
socket-timeout-millis		Specify the timeout for socket waiting for data in milliseconds.

Attribute	Default value	Description
ssl-required	**external**	Specify whether communication with the OpenID provider should be over HTTPS. The value can be one of the following: • **all** - all communication happens over HTTPS. • **external** - Only the communication with external clients happens over HTTPs. • **none** - HTTPs is not used.
token-signature-algorithm	**RS256**	Specify the token signature algorithm used by the OpenID provider. The supported algorithms are: • **RS256** • **RS384** • **RS512** • **ES256** • **ES384** • **ES512**
token-store		Specify cookie or session storage for auth-session data.
truststore		Specify the truststore used for client HTTPS requests.
truststore-password		Specify the truststore password.
verify-token-audience	**false**	If set to **true**, then during bearer-only authentication, the adapter verifies if token contains this client name (resource) as an audience.

Additional resources

- OpenID Connect configuration in JBoss EAP
- Securing a web application using OpenID Connect