

EJB3 over HTTP/HTTPS in JBossAS-5

This wiki describes how to access EJB3s via HTTP or HTTPS (HTTP over SSL). This is typically done when the beans are deployed behind a firewall so the client needs to communicate via a protocol and port allowed through the firewall. There are several steps required to configure the client, the server, and the beans to enable HTTP/HTTPS. Lets cover these one by one.

Note, the server should be shutdown when the following steps are being done.

Enabling Web Connectors

Edit the `%JBOSS_HOME%/server/servername/deploy/jbossweb.sar/server.xml` to enable (you need to uncomment the commented connector and point to the correct keystore file and provide the correct password) the HTTPS connector. Here's an example:

```
<!-- SSL/TLS Connector configuration using the admin devl guide keystore-->
<Connector protocol="HTTP/1.1" SSLEnabled="true"          port="8443"
address="{jboss.bind.address}"          scheme="https" secure="true" clientAuth="false"
      keystoreFile="{jboss.server.home.dir}/conf/myappkeystore.keystore"
keystorePass="change_this_password_appropriately" sslProtocol = "TLS" />
```

This enables the HTTPS connector on port 8443. The HTTP connector by default is enabled on port 8080.

Enabling Servlets

The next step is to enable the servlets which will be responsible for receiving the requests. To do this lets create a folder named `servlet-invoker.war` and place it in `%JBOSS_HOME%/`

server/servername/deploy folder. The servlet-invoker.war folder should contain a WEB-INF folder (case-sensitive) with a web.xml. The web.xml should be containing this:

```
<?xml version="1.0" encoding="UTF-8"?><!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> <servlet> <servlet-name>ServerInvokerServlet</servlet-
name> <description>The ServerInvokerServlet receives requests via
HTTP protocol from within a web container and passes it onto the
ServletServerInvoker for processing. </description> <servlet-
class>org.jboss.remoting.transport.servlet.web.ServerInvokerServlet</servlet-class>
<init-param> <param-name>locatorUrl</param-name> <param-
value>servlet://${jboss.bind.address}:8080/servlet-invoker/ServerInvokerServlet</param-
value> <description>The servlet server invoker</description> </init-param>
```

```
<load-on-startup>1</load-on-startup> </servlet>
```

```
<servlet> <servlet-name>SSLServerInvokerServlet</servlet-
name> <description>The ServerInvokerServlet receives requests via
HTTPS protocol from within a web container and passes it onto the
ServletServerInvoker for processing. </description> <servlet-
class>org.jboss.remoting.transport.servlet.web.ServerInvokerServlet</servlet-class>
<init-param> <param-name>locatorUrl</param-name> <param-
value>sslservlet://${jboss.bind.address}:8443/servlet-invoker/SSLServerInvokerServlet</
param-value> <description>The servlet server invoker</description> </init-
param>
```

```
<load-on-startup>2</load-on-startup> </servlet> <servlet-mapping> <servlet-
name>ServerInvokerServlet</servlet-name> <url-pattern>/ServerInvokerServlet/*</url-
pattern> </servlet-mapping>
```

```
<servlet-mapping> <servlet-name>SSLServerInvokerServlet</servlet-name>
<url-pattern>/SSLServerInvokerServlet/*</url-pattern> </servlet-mapping> </web-app>
```

This maps two servlets - one for http request and the other for https requests. These servlets are passed a "locatorUrl" init-param which should match the "InvokerLocator" attribute of the Remoting connectors that we create in the next step. This locatorURL value will also be used in the beans through the @RemoteBinding annotation's "clientBindUrl" property.

Enabling EJB3 Connectors

The next step is to deploy the remoting connectors for HTTP and HTTPS. Let's create a file name servlet-invoker-service.xml and drop it in %JBOSS_HOME%/server/servername/ deploy folder. The servlet-invoker-service.xml should contain this:

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

<server>

  <mbean code="org.jboss.remoting.transport.Connector"
name="jboss.remoting:service=connector,transport=servlet" display-
name="Servlet transport Connector"> <attribute name="InvokerLocator">servlet://
${jboss.bind.address}:8080/servlet-invoker/ServerInvokerServlet</
attribute> <attribute name="Configuration"> <handlers> <handler
subsystem="AOP">org.jboss.aspects.remoting.AOPRemotingInvocationHandler</
handler> </handlers> </attribute> </mbean>

  <mbean code="org.jboss.remoting.transport.Connector"
name="jboss.remoting:service=connector,transport=sslservlet" display-
name="Servlet transport Connector"> <attribute name="InvokerLocator">sslservlet://
${jboss.bind.address}:8443/servlet-invoker/SSLServerInvokerServlet</
attribute> <attribute name="Configuration"> <handlers> <handler
subsystem="AOP">org.jboss.aspects.remoting.AOPRemotingInvocationHandler</
handler> </handlers> </attribute> </mbean>

</server>
```

Here we configured two connectors - one for the "servlet" protocol and the other for "sslservlet" protocol. Notice that the "InvokerLocator" attribute of these connectors match the "locatorUrl" init-param for the servlets we configured in web.xml.

At this point you are done with the required configurations and can start your server and deploy the beans (see later section).

Keystores and Truststores

This section will provide a bit more details about the keystore and truststore files that you will require. You will need to generate public/private key pairs and digital certificates to enable SSL. The JDK provides a keytool utility for the generation and management of keys and certificates. The server keystore contains the server side public and private keys as well as the client's certificate, which includes the client public key. The server truststore contains the client's certificate, which indicates that the owner of the certificate is trusted. Conversely, the client side needs access to the truststore, which contains the server's certificate, which indicates that the owner of the certificate is trusted. Typically, the keystore and truststore are placed in the %JBASS_HOME%/server/servername/conf directory on the server side. Remember, we used this path in the server.xml earlier while enabling the HTTPS connector.

Bean Configuration

Now your EJB3 bean can use the following to binding the bean appropriately:

```
// This for the HTTPS@Stateless@RemoteBinding(clientBindUrl = "https://localhost:8443/  
servlet-invoker/  
SSLServerInvokerServlet")@Remote(Calculator.class)@SecurityDomain("other")public  
class CalculatorHttpsBean implements Calculator{....
```

```
// This is for HTTP@Stateless@RemoteBinding(clientBindUrl = "http://localhost:8080/servlet-  
invoker/ServerInvokerServlet")@Remote(Calculator.class)@SecurityDomain("other")public  
class CalculatorHttpBean extends CalculatorImpl{....
```

Client

The EJB3 client while looking up these beans should use the following properties for the JNDI lookup:

For HTTPS:

```
Properties props = new Properties();props.put("java.naming.factory.initial",
"org.jboss.naming.HttpNamingContextFactory");props.put("java.naming.provider.url",
"https://localhost:8443/invoker/JNDIFactory");props.put("java.naming.factory.url.pkgs",
"org.jboss.naming");Context ctx = new InitialContext(props);Calculator calculator =
(Calculator) ctx.lookup(jndiName);// use the bean to do any operations
```

For HTTP:

```
Properties props = new Properties();props.put("java.naming.factory.initial",
"org.jboss.naming.HttpNamingContextFactory");props.put("java.naming.provider.url",
"http://localhost:8080/invoker/JNDIFactory");props.put("java.naming.factory.url.pkgs",
"org.jboss.naming");Context ctx = new InitialContext(props);Calculator calculator =
(Calculator) ctx.lookup(jndiName);// use the bean to do any operations
```

While running the client, you need to pass the following truststore information:

```
java ..... -Djavax.net.ssl.trustStore=%JBOSS_HOME%/server/servername/conf/
myaptruststore.truststore -
```

EJB3 over HTTP/HTTPS in JBossAS-5

Djavax.net.ssl.trustStorePassword=change_this_password_appropriately -
Djava.protocol.handler.pkgs=javax.net.ssl