JBoss® Portal 2.7.2

SSO Frameworks Integration Guide

May 2009

1. Integration of JBoss Portal 2.7 with CAS 1
1.1. Integration of JBoss Portal and CAS deployed on the same host with usage of HTTP protocol 1
1.2. Integration of JBoss Portal and CAS deployed on the same host. HTTPS protocol is used for CAS 4
1.3. Integration of JBoss Portal and CAS deployed on the same host. HTTPS or HTTP protocol can be used for JBoss Portal 6
1.4. Integration of CAS and more JBoss Portals deployed on multiple hosts 8
1.5. Integration of CAS, JBoss Portal and thirdparty web application 12
1.6. Integration of one JBoss Portal and one CAS deployed on different hosts 14
1.7. FAQ 18
2. Integration of JBoss Portal 2.7 with JOSSO 21
2.1. Integration of JBoss Portal and JOSSO deployed on the same host with usage of HTTP protocol 21
2.2. Integration of JBoss Portal and JOSSO deployed on the same host. HTTPS protocol is used for JOSSO 27
2.3. Integration of JOSSO and more JBoss Portals deployed on multiple hosts 28
2.4. Integration of JOSSO, JBoss Portal and thirdparty web application 31
2.5. Integration of one JBoss Portal and one JOSSO server deployed on different hosts 32
2.6. FAQ 37

Integration of JBoss Portal 2.7 with CAS

Marek Posolda

This chapter describes how to integrate Jboss Portal 2.7 with CAS (Central Authentication Service). We will use version 3.0.7 of CAS and version 2.7.2 of JBoss Portal.

1.1. Integration of JBoss Portal and CAS deployed on the same host with usage of HTTP protocol

Note

This is the most simple scenario to configure. Single instance of JBoss Portal 2.7 is deployed on the same host and on the same instance of JBoss AS as CAS server. Only HTTP protocol is used in this scenario. So we are not interested about HTTPS protocol right now. This configuration shouldn't be used for production use (usage of HTTP protocol with CAS is not good idea). But the advantage is, that this configuration is really simple and this is good start for more advanced configurations.

1. You must have instance JBoss Portal 2.7 deployed on JBoss AS or JBoss EAP. Description of how to do this is out of scope of this document. You should read *Chapter 2* [http://docs.jboss.com/jbportal/v2.7.1/referenceGuide/html/installation.html] of reference guide if you are in trouble.

   In rest of this scenario, we assume that $JBOSS_HOME is the location when your JBoss AS is located and JBoss Portal is deployed on *default* configuration of your JBoss AS.

2. Download CAS (version 3.0.7). You can simply download it from *http://www.jasig.org/cas/download*. Once you have ZIP archive you can unpack it. You will find *cas.war* file in *target* directory of your unpacked ZIP archive. It is good idea to unpack this *cas.war* file to some location to your computer. We assume that you unpack it to */tmp/cas.war* directory in rest of this guide. The reason of this is necessity of configuring content of *cas.war* before we can deploy it to JBoss AS. And it's easier to edit content of directory than content of archive.

3. You have to copy two JAR libraries from *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/lib* directory to */tmp/cas.war/WEB-INF/lib* directory. The first file is *portal-identity-lib.jar* and second is *portal-identity-sso-lib.jar*.

4. Add this line:

```
<bean class="org.jboss.portal.identity.sso.cas.CASAuthenticationHandler" />
```

into file */tmp/cas.war/WEB-INF/deployerConfigContext.xml* instead of this line:

```
<bean class="org.jasig.cas.authentication.handler.support.
 SimpleTestUsernamePasswordAuthenticationHandler" />
```

Authentication handler is very important component of CAS server configuration, because it is used to authenticate users. If you will use *org.jboss.portal.identity.sso.cas.CASAuthenticationHandler*, you will be able to authenticate against CAS server with same credentials as against JBoss Portal, which is deployed on same host.

5. Add the CAS mbean configuration to *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/META-INF/jboss-service.xml* file. You have to add this to some place inside *server* element.

```
<!-- CAS -->
<mbean
  code="org.jboss.portal.identity.sso.cas.CASAuthenticationService"
  name="portal:service=Module,type=CASAuthenticationService"
  xmbean-dd=""
  xmbean-code="org.jboss.portal.jems.as.system.JBossServiceModelMBean">
  <xmbean/>
  <depends>portal:service=Module,type=IdentityServiceController</depends>
  <attribute name="HavingRole"></attribute>
</mbean>
```

6. Edit *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml*. Uncomment CAS valve and edit it as follows:

```
<Valve className="org.jboss.portal.identity.sso.cas.CASAuthenticationValve"
   casLogin="http://localhost:8080/cas/login"
   casLogout="http://localhost:8080/cas/logout"
```

```
    casValidate="http://localhost:8080/cas/serviceValidate"
    casServerName="localhost:8080"
    authType="FORM"
/>
```

Little description of attributes:

- *className:* className of authentication valve. It's very unprobable that you will be interested about changing this value to some other.
- *casLogin:* URL with CAS login screen. You will be redirected to this URL while click to login link in JBoss Portal.
- *casLogout:* URL with CAS logout screen. You will be redirected to this URL while click to logout link in JBoss Portal.
- *casValidate:* URL with CAS validation service. You will be redirected to this URL while you will try to submit authentication credentials in CAS login screen.
- *casServerName:* Host and port of application, which uses CAS. So this must be host and port where your JBoss Portal is deployed. While the previous attributes casLogin, casLogout and casValidate were related to CAS server, this attribute is related to JBoss Portal. In this scenario, the both hosts and ports are the same, so localhost:8080 is used on both places.

Note

You have to know that usage of *localhost* is suitable only for testing purposes. Because you won't be able to login from other location than localhost. You should use real name of your host in production environment.

7. Go to *http://repository.jboss.com/cas/3.0.7/lib/*. Download the file *casclient-lenient.jar* and copy it to *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/lib/* directory.

   Make sure to use *casclient-lenient.jar* and not *casclient.jar* in this step. You have to use *casclient-lenient.jar* if you use CAS server with HTTP protocol. Another reason is that you are using CAS server with HTTPS protocol but the SSL certificate you are using is not trusted by your Java implementation (or more exactly by the Java process under which JBoss AS with JBoss Portal is running). The more will be said later in this document.

8. Deploy cas.war into server. This can be donne simply by copying */tmp/cas.war* directory into *$JBOSS_HOME/server/default/deploy* directory.

9. Go to *$JBOSS_HOME/bin* directory and start the default configuration of JBoss AS.

10. When server is started, you can go to *http://localhost:8080/portal* and try to click to Login link. You shoud be redirected to CAS login screen in *http://localhost:8080/cas/login*. Now you are able to login with same user credentials as to the JBoss Portal itself without CAS integration. So you can try username/password:admin/admin or user/user if you are using fresh instance of JBoss Portal without any other users created. After successfull login, you should be redirected to JBoss Portal default page and you should see sign *Logged in as: admin* or *Logged in as: user* in right up corner.

1.2. Integration of JBoss Portal and CAS deployed on the same host. HTTPS protocol is used for CAS

Note

This scenario will show you how to secure your JBoss AS instance with CAS server and how to authenticate to CAS in secure way with usage of HTTPS protocol. You should always use HTTPS protocol for CAS server in production environment. JBoss Portal itself uses only HTTP protocol in this scenario. The assumption is that you did the previous scenario *1.1* and now you have CAS and JBoss Portal on the same host and you are using HTTP protocol for CAS authentication.

1. Now you must secure your JBoss AS instance with CAS (The default instance in our case) if you want to use HTTPS protocol. The first think you need is the SSL certificate your server will use. For production use, it's very suitable if you have SSL certificate from some well-known certification authority, especially if you want to use your host in public environment (not only Intranet in your company). For testing purposes is sufficient that we have only self-signed certificate which can be created by simple tool called *keytool*. This tool is standard part of Sun Java JDK environment.

   So go to your *$JBOSS_HOME/server/default/conf* directory and generate self-signed certificate by running command:

```
keytool -genkey -alias jbosskey -keypass changeit -keyalg RSA -keystore server.keystore
```

   You have to use password *changeit* as the answer for the first question. It's good that the keystore password is the same like key password, which is specified by option *keypass* in command line. The second question is *What is your first and last name* and you should write the fully-qualified name of your host as answer to this question. For our purposes, it's sufficient if you use value *localhost*. Other values are not important for our testing purposes and you

can fill defacto anythink you want. After all questions are answered by you and the command is finished, you should see file *server.keystore* in your *$JBOSS_HOME/server/default/conf* directory.

2. You have to enable HTTPS connector for JBoss AS. You can do this by editing file *$JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml*. Uncomment the part with HTTPS connector and edit it as follows:

```
<Connector port="8443" maxHttpHeaderSize="8192" address="${jboss.bind.address}"
        maxThreads="250" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" disableUploadTimeout="true"
        acceptCount="100" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
        SSLEnabled="true"
        keystoreFile="${jboss.server.home.dir}/conf/server.keystore"
        keystorePass="changeit"
/>
```

3. Now you can restart your JBoss AS and then going to *https://localhost:8443*. You will see the warning that the certificate is not trusted. It's fine because you are using your self-signed certificate and your browser doesn't know this certificate. After accepting all warnings you should see the default JBoss AS page and now you are able to secure your CAS server. You should avoid to use self-signed certificates in production environment, because all user see these warnings in their browsers and they are not due to trust you.

   If you have any troubles with SSL, you want to use certificate from well-known certification authority or you want to know more about SSL, you can go to

   - *http://www.jboss.org/community/wiki/SSLSetup* - How to configure SSL in JBoss AS
   - *http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html* - How to configure SSL in Tomcat. It's very useful guide with the most important informations about SSL.
   - *http://en.wikipedia.org/wiki/Transport_Layer_Security* - Very detailed description from wikipedia.

4. Configure CASAuthenticationValve in *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml* file. Edit it as follows:

```
<Valve className="org.jboss.portal.identity.sso.cas.CASAuthenticationValve"
   casLogin="https://localhost:8443/cas/login"
   casLogout="https://localhost:8443/cas/logout"
   casValidate="https://localhost:8443/cas/serviceValidate"
   casServerName="localhost:8080"
   authType="FORM"
/>
```

You can see that we use HTTPS on port 8443 for all URLs related to CAS. But we are still using *localhost:8080* for casServerName. So we will use HTTP while going to JBoss Portal. Only access to CAS is secured.

5. You should use normal *casclient.jar* instead of *casclient-lenient.jar* because we are now using HTTPS with CAS. So go to *http://repository.jboss.com/cas/3.0.7/lib/* and download *casclient.jar*. Copy this file to *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/lib* directory. Don't forget to remove *casclient-lenient.jar* from this directory too.

6. Now you should restart your JBoss AS, but you have to run it with special option -*Djavax.net.ssl.trustStore*. This Java option is really important because you are saying where is file with certificates, which should be trusted by Java. When your JBoss Portal needs to trust CAS, it's important that JBoss AS instance with JBoss Portal trusts CAS server and so we need to say that our self-signed certificate is trustfull.

   The whole command for starting JBoss should look like this:

```
./run.sh -Djavax.net.ssl.trustStore=$JBOSS_HOME/server/default/conf/server.keystore
```

7. When server is started, you can go to *http://localhost:8080/portal* and try to click to Login link. You shoud be redirected to CAS login screen in *https://localhost:8443/cas/login*. After successfull login, you should be redirected to JBoss Portal default page on *http://localhost:8080/portal/auth/portal/default* and you should see sign *Logged in as: admin* or *Logged in as: user* in right up corner.

1.3. Integration of JBoss Portal and CAS deployed on the same host. HTTPS or HTTP protocol can be used for JBoss Portal

When you configure JBoss Portal and CAS in the scenario described in *1.2* and you try to go to *https://localhost:8443/portal* then if you login succesfully with CAS, you won't be redirected to JBoss Portal. It's caused because you are redirected to URL like *https://localhost:8080/portal*, which is invalid form of URL (you try to use HTTPS with port 8080).

You can repair this by configuring CASAuthenticationValve in *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml* and change the value of *casServerName* to *localhost:8443* instead of *localhost:8080*. Now you can simply go to *https://localhost:8443/portal* and login yourself, but you can't simply login while going through JBoss Portal with HTTP. So what to do if you want to use both HTTP and HTTPS protocols to access JBoss Portal?

The simplest way is to use standard ports for both services HTTP and HTTPS. So you needn't specify port in *casServerName* but only host. The disadvantage is, that you must have root privileges in Linux systems or admin privileges in Windows. Because only root user can run processes which listens on less port numbers than 1024.

1. Change ports of both connectors in *$JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml*. HTTP connector should look like this:

```
<Connector port="80" address="${jboss.bind.address}"
   maxThreads="250" maxHttpHeaderSize="8192"
   emptySessionPath="true" protocol="HTTP/1.1"
   enableLookups="false" redirectPort="443" acceptCount="100"
   connectionTimeout="20000" disableUploadTimeout="true"
/>
```

   and HTTPS connector like this:

```
<Connector port="443" maxHttpHeaderSize="8192" address="${jboss.bind.address}"
 maxThreads="250" minSpareThreads="25" maxSpareThreads="75"
 enableLookups="false" disableUploadTimeout="true"
 acceptCount="100" scheme="https" secure="true"
 clientAuth="false" sslProtocol="TLS"
 SSLEnabled="true"
 keystoreFile="${jboss.server.home.dir}/conf/server.keystore"
 keystorePass="changeit"
/>
```

2. Change configuration in *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml* and edit the CAS Authentication Valve as follows:

```
<Valve className="org.jboss.portal.identity.sso.cas.CASAuthenticationValve"
   casLogin="https://localhost/cas/login"
   casLogout="https://localhost/cas/logout"
   casValidate="https://localhost/cas/serviceValidate"
   casServerName="localhost"
   authType="FORM"
/>
```

3. Try to restart JBoss AS. Don't forget that you must be root to this step. In linux system you can do it with this command:

```
sudo ./run.sh -Djavax.net.ssl.trustStore="$JBOSS_HOME/server/default/conf/server.keystore"
```

4. You can use JBoss Portal with HTTPS and go to *https://localhost/portal* and then login. But now you can use HTTP protocol too and you can successfully login with URL *http://localhost/portal*. HTTPS protocol is always used for CAS but both HTTP and HTTPS protocols can be used for JBoss Portal if it is configured this way.

1.4. Integration of CAS and more JBoss Portals deployed on multiple hosts

The main advantage of SSO is possibility to use one authentication system against many different applications. And with CAS we can use this feature in very nice form. If we have more applications configured against CAS, than after login to one application, we are authenticated to other applications too. No matter if these applications are on different hosts. The only conditions are, that these applications must be configured against same CAS and you use the same browser for authenticating with cookies enabled.

When you are trying to authenticate to some application called *webapp1* with usage of CAS, than you are redirected to CAS login page. After successfull authentication, the CAS cookie is set in CAS web application and the ticket is granted to you to access *webapp1*. When you try to login to other application *webapp2* you are redirected to CAS login page again, but when the cookie was set, you are automatically authenticated with the same CAS receipt as in the first application. So you are automatically redirected to your *webapp2*.

Note

Important condition is that you must have cookies enabled in your browser if you want to use CAS this way!

Next scenario simulates situation when two instances of JBoss Portal are deployed on different hosts and both are configured to authenticate against one CAS server. In this tutorial, we will simulate this scenario simply on one computer but we will use different configurations of JBoss AS (EAP) and we will use different host names for each configuration. In later text, the *cas-01* will be the name of the host and JBoss AS configuration where we will have JBoss Portal and CAS deployed. The *cas-02* will be the name of the host and JBoss AS configuration where we will have second JBoss Portal deployed. This second portal will be configured for authentication against CAS on cas-01.

1. Create new virtual hosts on your system and use some different names for them. In this scenario, We willl use names *cas-01* and *cas-02*. On linux systems, it can be donne by add these two lines

```
127.0.1.4       cas-01
127.0.1.5       cas-02
```

   to your */etc/hosts* file. You must have root privileges to do this. If you don't have root privileges, than you have to use IP addresses instead of host names in rest of this tutorial (For example: Running of JBoss Portal can be performed by *./run.sh -c cas-01 -b 127.0.1.4* instead of *./run.sh -c cas-01 -b cas-01*. You must use IP addresses in your configuration files too.).

2. Create JBoss AS configuration called *cas-01* and deploy JBoss Portal and CAS to it. We pressume that you did previous scenario *1.3* and you have JBoss Portal with CAS properly configured in your default configuration. So only think you have to do is copy your *default* configuration to new *cas-01* configuration. This can be donne with command

```
cp -r $JBOSS_HOME/server/default $JBOSS_HOME/server/cas-01
```

   on linux systems.

3. Create JBoss AS configuration called *cas-02* and deploy JBoss Portal to it. You can do it by copying *cas-01* configuration to new *cas-02* configuration and then you can undeploy CAS from *cas-02* configuration because you doesn't need CAS server in *cas-02*. JBoss Portal from *cas-02* will be configured against CAS server on *cas-01*. You can do both thinks on linux systems this way:

```
cp -r $JBOSS_HOME/server/cas-01 $JBOSS_HOME/server/cas-02
rm -rf $JBOSS_HOME/server/cas-02/deploy/cas.war
```

4. You should generate new self-signed certificate for host *cas-01* because previous certificate was CN attribute with value *localhost*. Now we will use host *cas-01* for this configuration, so we have to generate new certificate with CN value cas-01.

   So go to directory *$JBOSS_HOME/server/cas-01/conf* and delete file *server.keystore* if one is here. Now you should generate new certificate with usage of command

```
keytool -genkey -alias jbosskey -keypass changeit -keyalg RSA -keystore server.keystore
```

   Answer *changeit* as value for keystore password and answer *cas-01* as answer for question about your first and second name. Other answers are not so important.

5. Go to *$JBOSS_HOME/server/cas-02/conf* directory and do the same with certificate for *cas-02* host. CN attribute should have value *cas-02* for this configuration.

6. Configure CAS Authentication Valve in *cas-01* JBoss AS configuration. You can do it by edit *$JBOSS_HOME/server/cas-01/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml* this way:

```
<Valve className="org.jboss.portal.identity.sso.cas.CASAuthenticationValve"
   casLogin="https://cas-01/cas/login"
   casLogout="https://cas-01/cas/logout"
   casValidate="https://cas-01/cas/serviceValidate"
   casServerName="cas-01"
```

```
    authType="FORM"
/>
```

The important think is, that your HTTP connector must listen on port 80 and your HTTPS connector on port 443 if you want to configure CAS server this way. Look to *1.3* if you are not sure how to configure ports correctly. If you did *1.3* scenario before, than you have ports configured correctly and you won't care about it.

7. Configure CAS Authentication Valve in *cas-02* JBoss AS configuration. You can do it by edit *$JBOSS_HOME/server/cas-02/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml* this way:

```
<Valve className="org.jboss.portal.identity.sso.cas.CASAuthenticationValve"
    casLogin="https://cas-01/cas/login"
    casLogout="https://cas-01/cas/logout"
    casValidate="https://cas-01/cas/serviceValidate"
    casServerName="cas-02"
    authType="FORM"
/>
```

8. Start JBoss AS with *cas-01* configuration and bind it to host *cas-01*.

```
sudo ./run.sh -c cas-01 -b cas-01 -Djavax.net.ssl.trustStore="$JBOSS_HOME/server/cas-01/conf/server.keystore"
```

9. Start JBoss AS with *cas-02* configuration and bind it to host *cas-02*.

```
sudo ./run.sh -c cas-02 -b cas-02 -Djavax.net.ssl.trustStore="$JBOSS_HOME/server/cas-01/conf/server.keystore"
```

```
```

The parameter *javax.net.ssl.trustStore* remains in the same location as in *cas-01* because you have to trust to SSL certificate which CAS server is using. And that is the certificate from *cas-01* configuration.

10. Try SSO in action. Go to *http://cas-01/portal* and click to login link. You should be redirected to CAS login page on *http://cas-01/cas/login*. After login and validating of your credentials, you should be logged and after that you are redirected back to portal on *http://cas-01/portal/auth/portal/default*. Now try second JBoss Portal on *cas-02* host. Go to *http://cas-02/portal* and click to login link. You are redirected directly to *http://cas-02/portal/auth/portal/default* without need of filling any credentials. You can see that you are logged on *cas-02* with same user as on *cas-01*. The CAS will grant ticket to second portal automatically because CAS cookie is set with information, that you were logged before.

1.5. Integration of CAS, JBoss Portal and thirdparty web application

As I said before in *1.4*, you can use CAS for authentication to more web applications on different hosts. In next scenario we will show situation, when we use CAS with some sample thirdparty web application. After we succesfully login to that application with CAS, you will see that we are logged to the JBoss Portal too.

The assumption is that you have donne *previous scenario* and so you have two hosts *cas-01* and *cas-02*. CAS is deployed on *cas-01*.

1. Download *CASIntegrationExampleApp* from URL ...TODO... and go to the dir with this application.
2. You have to edit *build.properties* file according to your path where you have JBoss AS. Assuming that you will use *cas-02* JBoss configuration to deploy the application. So you have property *jboss.app.deploy.dir* configured as follows:

```
jboss.app.deploy.dir=${jboss.home}/server/cas-02/deploy
```

3. *CASIntegrationExampleApp* is very small web application with only one servlet. Only important think is, that this application is configured to use CAS for authentication. The configuration is performed by adding CAS Filer to the *CASIntegrationExampleApp/WEB_INF/web.xml* file. The whole configuration of CAS Filter looks as follows:

```
<filter>
 <filter-name>CAS Filter</filter-name>
 <filter-class>edu.yale.its.tp.cas.client.filter.CASFilter</filter-class>
 <init-param>
  <param-name>edu.yale.its.tp.cas.client.filter.loginUrl</param-name>
  <param-value>https://cas-01/cas/login</param-value>
 </init-param>
 <init-param>
  <param-name>edu.yale.its.tp.cas.client.filter.validateUrl</param-name>
  <param-value>https://cas-01/cas/serviceValidate</param-value>
 </init-param>
 <init-param>
  <param-name>edu.yale.its.tp.cas.client.filter.serverName</param-name>
  <param-value>cas-02</param-value>
 </init-param>
 <init-param>
  <param-name>edu.yale.its.tp.cas.client.filter.wrapRequest</param-name>
  <param-value>true</param-value>
 </init-param>
</filter>

<filter-mapping>
 <filter-name>CAS Filter</filter-name>
 <url-pattern>/*</url-pattern>
</filter-mapping>
```

Little description of init parameters:

- *edu.yale.its.tp.cas.client.filter.loginUrl:* URL with CAS login screen. You will be redirected to this URL while trying to authenticate.
- *edu.yale.its.tp.cas.client.filter.validateUrl:* URL with CAS validation service. You will be redirected to this URL while you will try to submit authentication credentials in CAS login screen.
- *edu.yale.its.tp.cas.client.filter.serverName:* Host and port where you will be redirected after successfull authentication. So the same host where *CASIntegrationExampleApp* will be deployed should be specified here.

- *edu.yale.its.tp.cas.client.filter.wrapRequest:* If value of this parameter is true, than calling of *httpServletRequest.getRemoteUser()* should return the name of authenticated user if you use it in your web application.

You can casify your own applications in very similar way. If you are more interested about configuration of CAS Filter, you can go to *http://www.ja-sig.org/wiki/display/CASC/Using+CASFilter*.

The class *edu.yale.its.tp.cas.client.filter.CASFilter* is part of *casclient.jar* which we used in previous part of this document. Make sure that you have access to this library in your applications.

4. If you are in directory with *CASInterationExampleApp* you have to call

```
ant all
```

and the application will be deployed to *cas-02* server.

5. Once application is deployed, you can go to *http://cas-02/CASIntegrationExampleApp* and you should be redirected to CAS login screen. You have to login with same credentials as to Jboss Portal in *cas-01* server. Once you are logged, you will be redirected to sample servlet and the name of the authenticated user will be shown.

   Go to *http://cas-02/portal* and click to login link. Now you can see that you are authenticated directly to JBoss Portal without redirecting to CAS.

   Go to *http://cas-01/portal* and click to login link. You should be redirected directly to this portal too.

1.6. Integration of one JBoss Portal and one CAS deployed on different hosts

The last scenario describes situation, when we want to deploy JBoss Portal in a host *portal-host* and we want to deploy CAS server to different host *cas-host*. Configuration of this scenario is different in many ways because we can't use *org.jboss.portal.identity.sso.cas.CASAuthenticationHandler* as authentication handler during this scenario. The reason is that *CASAuthenticationHandler* uses *CASAuthenticationService* and this service depends on *IdentityServiceController*, which is the base component of identity in JBoss Portal and authentication depends on it.

If you configure CAS to use *org.jboss.portal.identity.sso.cas.CASAuthenticationHandler* then you are able to authenticate to CAS with same user credentials, which are used by the instance of JBoss Portal, which is deployed on the same server as CAS server. But if you want to use org.jboss.portal.identity.sso.cas.CASAuthenticationHandler and there is not JBoss Portal deployed on the host with CAS, then you end with exception while trying to validate user credentials from CAS login screen.

In this scenario we will configure JBoss Portal with database and we will configure CAS to access directly the same database. CAS comes with set of standard authentication handlers and we will use one of them to bind Mysql database.

1. Create new virtual hosts on your system and use some different names for them. In this scenario, We will use names *portal-host* and *cas-host*. On linux systems, it can be donne by add these two lines

```
127.0.1.6       portal-host
127.0.1.7       cas-host
```

   to your */etc/hosts* file. You must have root privileges to do this. If you don't have root privileges, than you have to use IP addresses instead of host names in rest of this tutorial. In later text, cas-host will be a host where CAS server will be deployed and portal-host will be the host where JBoss Portal will be deployed.

2. Take a fresh JBoss AS or JBoss EAP (for example JBoss-4.2.3.GA is a good). Then create new configuration called *cas-conf* where CAS will be deployed. You can use this command in linux:

```
cp -r $JBOSS_HOME/server/default $JBOSS_HOME/server/cas-conf
```

3. Deploy JBoss Portal 2.7.2 to the default configuration of JBoss AS. We assume that you use Mysql database on localhost. So your datasource could look like this:

```
```

```
<datasources>
  <local-tx-datasource>
    <jndi-name>PortalDS</jndi-name>
    <connection-url>jdbc:mysql://localhost/jboss_portal?jdbcCompliantTruncation=false
    </connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>jboss_portal</user-name>
    <password>password</password>
  </local-tx-datasource>
</datasources>
```

You can use any other database than Mysql. Only important think is, that you must configure CAS against the same database as JBoss Portal in later steps of this scenario. You can read *Chapter 2* [http://docs.jboss.com/jbportal/v2.7.1/referenceGuide/html/installation.html] of portal reference guide if you have some troubles with deploying JBoss Portal configured against custom datasource.

4. Configure CAS authentication valve in *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml*

```
<Valve className="org.jboss.portal.identity.sso.cas.CASAuthenticationValve"
      casLogin="http://cas-host:8080/cas/login"
      casLogout="http://cas-host:8080/cas/logout"
      casValidate="http://cas-host:8080/cas/serviceValidate"
      casServerName="portal-host:8080"
      authType="FORM"
/>
```

5. Download *casclient-lenient.jar* from *http://repository.jboss.com/cas/3.0.7/lib/* and copy it to *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/lib/* directory. If you want to use *casclient.jar* you have to configure CAS with usage of HTTPS. Go to *1.2* scenario for more informations.

6. We assume that you have fresh instance of CAS in */tmp/cas.war* directory. You have to add following jars to */tmp/cas.war/WEB-INF/lib* directory.

   - *commons-dbcp-1.2.jar* - You can download it from *http://mirrors.ibiblio.org/pub/mirrors/maven2/commons-dbcp/commons-dbcp/1.2/commons-dbcp-1.2.jar*.

- *commons-pool-1.3.jar* - You can download it from *http://mirrors.ibiblio.org/pub/mirrors/maven2/commons-pool/commons-pool/1.3/commons-pool-1.3.jar*.
-
- *cas-server-jdbc-3.0.7.jar* - You can download *http://www.ja-sig.org/downloads/cas/cas-server-3.0.7.zip* and the file *cas-server-jdbc-3.0.7.jar* is iside of *cas-server-3.0.7/target* directory of the archive.
- *mysql-jdbc.jar* - You can download it from mysql page. You have to use different JDBC driver according to database you are using.

7. Configure CAS against the same database which is used by JBoss Portal. The information about portal users is standardly saved in table *jbp_users*. Column with username is *jbp_uname* and column with password is *jbp_password*. Passwords are hashed by MD5 algorithm.

   You have to open */tmp/cas.war/WEB-INF/deployerConfigContext.xml* and configure it properly:

   - Configure datasource and put it directly inside the *beans* element.

```
<bean id="mysql_dataSource" class="org.apache.commons.dbcp.BasicDataSource">
   <property name="driverClassName">
     <value>com.mysql.jdbc.Driver</value>
   </property>
   <property name="url">
     <value>jdbc:mysql://localhost:3306/jboss_portal</value>
   </property>
   <property name="username"><value>jboss_portal</value></property>
   <property name="password"><value>password</value></property>
</bean>
```

   - Configure authentication handler. Put this:

```
<bean class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler">
        <property name="dataSource" ref="mysql_dataSource" />
            <property name="sql" value="SELECT jbp_password FROM jbp_users WHERE
 jbp_uname=?" />
        <property name="passwordEncoder">
```

```
            <bean class="org.jasig.cas.authentication.handler.Md5PasswordEncoder" />
        </property>
</bean>
```

to the file instead of

```
<bean class="org.jasig.cas.authentication.handler.support.
 SimpleTestUsernamePasswordAuthenticationHandler" />
```

8. Deploy CAS into *cas-conf* configuration.

```
cp -r /tmp/cas.war $JBOSS_HOME/server/cas-conf/deploy/
```

9. Start both configurations of JBoss AS and bind them to different hosts. First the default configuration with JBoss Portal:

```
./run.sh -c default -b portal-host
```

and then *cas-conf* configuration with CAS server.

```
./run.sh -c cas-conf -b cas-host
```

10. Test integration. Go to *http://portal-host:8080/portal* and try to login. You should be redirected to CAS server *http://cas-host:8080/cas/login* and you should be able to login with default portal credentials.

1.7. FAQ

Which versions of JBoss Portal and CAS should I use?

All scenarios are tested with JBoss Portal 2.7.2 branch version (*http://anonsvn.jboss.org/repos/portal/branches/JBoss_Portal_Branch_2_7* [http://anonsvn.jboss.org/repos/portal/branches/JBoss_Portal_Branch_2_7/]) deployed on JBoss AS 4.2.3.GA. Version of CAS is 3.0.7. Configurations can be a little different if you use different versions of components.

I did scenario *1.4* and I created user "johnny" in cas-02 portal. But when I try to login with CAS to access cas-02 portal, I am not able to login as johnny

Yes. It works this way. Because our CAS server on cas-01 is configured to use *org.jboss.portal.identity.sso.cas.CASAuthenticationHandler*. This authentication handler uses *CASAuthenticationService* and this service depends on *IdentityServiceController*, which is the base component of authentication to JBoss Portal. So if you configure CAS to use *CASAuthenticationHandler* then you are able to authenticate to CAS with same user credentials, which are used by the instance of JBoss Portal, which is deployed on the same server as CAS server. So if you create user *johnny* in JBoss Portal on cas-01 portal, then everythink is fine and you are able to login as this user.

Try this scenario:

- Go to cas-01 portal and create user *johnny* with password *cas01_password*. Give him only Users role.
- Go to cas-01 portal and create user *johnny* with password *cas02_password*. Give him both Users and Administrators role.
- Try to authenticate as johnny with CAS. You must use *cas01_password*. But when you go to JBoss Portal on cas-02, you can see that johnny has admin privileges. It works this way because CAS is used only for authentication of users, but authorization is in scope of each JBoss Portal. So roles from cas-02 are used for johnny while he is in portal on cas-02. But when you will try to go to cas-01 as johnny, you should be only in user role.

I have JBoss Portal configured against LDAP and I want to use CAS for authentication deployed on different host

You can use one from LDAP authentication handlers. Or you can implement your own authentication handler if none from standard handlers is suitable for your needs. Useful links:

- *http://www.ja-sig.org/wiki/display/CASUM/LDAP* - LDAP authentication handlers on CAS Wiki
- *http://www.jasig.org/cas/server-deployment/authentication-handler* - How to write authentication handler

I have JBoss Portal integrated with CAS. But CAS server is unreachable right now

That's bad because when you attempt to login, then CAS authentication valve redirects you to cas login screen. And if this screen is unreachable you end with 404 error. So you are not able to login if CAS server is out.

Is it possible to avoid using of javax.net.ssl.trustStore parameter while starting JBoss?

Yes but your Java implementation must trust to the certificate of server where CAS is deployed. Default truststore file is in your Java home directory. Exactly in *$JAVA_HOME/jre/lib/security/cacerts* file. So you must import your certificate to this file. Go to your $JBOSS_HOME/server/cas-01/conf directory. And run commands:

```
keytool -export -alias jbosskey -keypass changeit -keystore server.keystore -file server.crt
keytool -import -trustcacerts -file server.crt -keypass changeit -alias jbosskey -keystore $JAVA_HOME/jre/lib/security/cacerts
```

Now you should be able to run your cas-01 server with command

```
sudo ./run.sh -c cas-01 -b cas-01
```

How the JBoss Portal behaves when session timeout occurs?

When session timeout occurs in JBoss Portal, then session is invalidated. But after click to some link in JBoss Portal, the user is directly logged in without need to fill any credentials. The reason is, that if CAS cookie is set, then CAS gives you a new ticket and you are not forced to fill credentials again. So you are directly logged even if session attributes (and all portlet session attributes from portlet scope or application scope) are deleted.

Integration of JBoss Portal 2.7 with JOSSO

Marek Posolda

This chapter describes how to integrate Jboss Portal 2.7 with SSO framework JOSSO (Java Open Single Sign On). We will use version 1.8.0 of JOSSO and version 2.7.2 of JBoss Portal.

2.1. Integration of JBoss Portal and JOSSO deployed on the same host with usage of HTTP protocol

This is the basic scenario of integration. Both JBoss Portal and JOSSO will be deployed to the same instance of JBoss AS 4.2.X and they will be configured, so the JBoss Portal will use JOSSO for authentication and you will be able to login to JOSSO with same credentials as to JBoss Portal before integration.

1. You must have JBoss Portal 2.7 deployed on JBoss AS (version 4.2.2 or 4.2.3) or JBoss EAP. Description of how to do this is out of scope of this document. You should read *Chapter 2* [http://docs.jboss.com/jbportal/v2.7.1/referenceGuide/html/installation.html] of reference guide if you are in trouble.

   In rest of this scenario, we assume that $JBOSS_HOME is the location when your JBoss AS is located and JBoss Portal is deployed on *default* configuration of your JBoss AS.

2. Start JBoss server and stop it after the full start. This is necessary because of correct initialization of CMS.

3. Download *josso-1.8.0.zip* from *http://www.josso.org* and unzip it to some location on your computer. We assume */tmp/josso-1.8.0* in rest of this document.

4. Install JOSSO gateway and JOSSO agent to your *default* configuration of JBoss Portal. Gateway is the core component of JOSSO which acts as the web console and makes the authentication. Gateway is identity provider. JOSSO agent is somethink like listener, which redirects authentication requests to JOSSO gateway. Agent needs to be deployed in same instance like JBoss Portal.

   In our case we use only one configuration for both JBoss Portal and JOSSO. So both gateway and agent will be deployed to the *default* configuration. Installation could be performed with *josso-gsh*, which you can run by command:

```
/tmp/josso-1.8.0/bin/josso-gsh
```

Now you can install gateway and agent to your JBoss AS with command:

```
gateway install --target $JBOSS_HOME --jboss-instance default --platform jb42
agent install --target $JBOSS_HOME --jboss-instance default --platform jb42
```

It's possible that you see error about missing DTD document during agent installation. But you can ignore this error. The important assumption is that you have JBoss AS version 4.2.

5. You have to copy two JAR libraries from *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/lib* directory to *$JBOSS_HOME/server/default/deploy/josso.war/WEB-INF/lib* directory. The first file is *portal-identity-lib.jar* and second is *portal-identity-sso-lib.jar*.

6. Uncomment JOSSO logout valve from *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/context.xml*:

```
<Valve className="org.jboss.portal.identity.sso.josso.JOSSOLogoutValve"/>
```

7. Edit *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/META-INF/jboss-service.xml* and add JOSSOIdentityService mbean to configuration:

```
<!-- JOSSO -->
<mbean
   code="org.jboss.portal.identity.sso.josso.JOSSOIdentityServiceImpl"
   name="portal:service=Module,type=JOSSOIdentityService"
   xmbean-dd=""
   xmbean-code="org.jboss.portal.jems.as.system.JBossServiceModelMBean">
   <xmbean/>
   <depends>portal:service=Module,type=IdentityServiceController</depends>
</mbean>
```

8. Edit *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/login.jsp*. This file should be used for redirecting to JOSSO and only this content should be inside of this file:

```
<%@page contentType="text/html; charset=iso-8859-1" language="java" session="true" %>
<%
   response.sendRedirect(request.getContextPath() + "/josso_login/");
%>
```

Other stuff could be commented or deleted.

9. Add this application policy to the file *$JBOSS_HOME/server/default/conf/login-config.xml*:

```
<application-policy name="josso">
  <authentication>
                <login-module    code="org.jboss.portal.identity.sso.josso.JOSSOLoginModule"
 flag="required">
       <module-option name="debug">true</module-option>
     </login-module>
  </authentication>
</application-policy>
```

10. Change the portal security domain to josso in *$JBOSS_HOME/server/default/deploy/jboss-portal.sar/portal-server.war/WEB-INF/jboss-web.xml* by edit the line with security domain:

```
<security-domain>java:jaas/josso</security-domain>
```

11. Add the file *$JBOSS_HOME/server/default/conf/josso-gateway-portal-stores.xml* with this content:

```
<s:beans xmlns:s="http://www.springframework.org/schema/beans"
      xmlns:portal-istore="urn:org:jboss:portal:josso:identitystore"
      xmlns:memory-sstore="urn:org:josso:memory:sessionstore"
      xmlns:memory-astore="urn:org:josso:memory:assertionstore"
```

```
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
       http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-2.5.xsd
                urn:org:josso:memory:sessionstore  http://www.josso.org/schema/josso-memory-
sessionstore.xsd
              urn:org:josso:memory:assertionstore  http://www.josso.org/schema/josso-memory-
assertionstore.xsd
       ">
                                                                                <!--
 ========================================================================== -
->
   <!-- JOSSO Identity Store, the id is very important because it is          -->
   <!-- referenced by the identity manager, auth schemes and who knows where  -->
   <!-- else.                                                                 -->
                                                                                <!--
 ========================================================================== -
->
   <portal-istore:portal-store id="josso-identity-store" s:scope="singleton"/>

                                                                                <!--
 ========================================================================== -
->
   <!-- JOSSO Session Store, the id is very important because it is           -->
   <!-- referenced by the session manager and who knows where else            -->
                                                                                <!--
 ========================================================================== -
->
   <memory-sstore:memory-store id="josso-session-store"/>

                                                                                <!--
 ========================================================================== -
->
   <!-- JOSSO Assertion Store, the id is very important because it is         -->
   <!-- referenced by the assertion manager and who knows where elese         -->
                                                                                <!--
 ========================================================================== -
->
   <memory-astore:memory-store id="josso-assertion-store"/>
</s:beans>
```

12 Edit *$JBOSS_HOME/server/default/conf/josso-gateway-config.xml* file. It's the main configuration file of JOSSO gateway. You have to change two thinks in the file.

- Change the stores part of the file. Add josso-gateway-portal-stores and comment other stores:

```
<!-- Identity, Session and Assertion Stores configuration -->
<s:import resource="josso-gateway-portal-stores.xml" />
<!--
<s:import resource="josso-gateway-memory-stores.xml" />
<s:import resource="josso-gateway-db-stores.xml" />
<s:import resource="josso-gateway-ldap-stores.xml" />
-->
```

- Bind authentication scheme needs to be registered in another part of this file (others can be commented out):

```
<def-auth:authenticator id="josso-authenticator">
  <def-auth:schemes>
    <s:ref bean="josso-bind-authentication"/>
    <!--
    <s:ref bean="josso-basic-authentication"/>
    <s:ref bean="josso-strong-authentication"/>
    <s:ref bean="josso-rememberme-authentication"/>
    -->
    <!-- Others like NTLM and BIND go here -->
    <!--
    <s:ref bean="josso-bind-authentication"/>
    -->
  </def-auth:schemes>
</def-auth:authenticator>
```

13 Edit *$JBOSS_HOME/server/default/conf/josso-gateway-auth.xml* and uncomment bind authentication scheme:

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- BIND Authentication Scheme (normally LDAP) -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- Requires a be a bindalble credential store ! -->
<!-- name attribute is important and must not be changed -->
```

```
<bind-authscheme:bind-auth-scheme
        id="josso-bind-authentication"
        name="basic-authentication"
        hashAlgorithm="MD5"
        hashEncoding="HEX"
        ignorePasswordCase="false"
        ignoreUserCase="false">


    <bind-authscheme:credentialStore>
        <s:ref bean="josso-identity-store"/>
    </bind-authscheme:credentialStore>

    <bind-authscheme:credentialStoreKeyAdapter>
        <s:ref bean="josso-simple-key-adapter"/>
    </bind-authscheme:credentialStoreKeyAdapter>
</bind-authscheme:bind-auth-scheme>
```

14 Edit *$JBOSS_HOME/server/default/conf/josso-agent-config.xml* and edit it for usage with JBoss Portal application. You should have this content inside your <agent:partner-apps> element.

```
<agent:partner-apps>
  <agent:partner-app id="jboss_portal" context="/portal"/>
</agent:partner-apps>
```

All other partner-apps can be commented out or deleted unless you want to use other applications with JOSSO.

15 Add *-Dorg.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl* option to your JAVA_OPTS environment variable. The most simple way is to add the line

```
JAVA_OPTS="$JAVA_OPTS -Dorg.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl"
```

to your *$JBOSS_HOME/bin/run.sh* file. (Some place before *Display our enviroument* section)

16. Start JBoss again and now you should be able to test the integration. Go to *http://localhost:8080/portal* and click to login link. You should be redirected to josso gateway web console. You should be able to login with your portal credentials (for example: admin/admin or user/user). After login, you should be redirected back to Portal and be logged into portal.

2.2. Integration of JBoss Portal and JOSSO deployed on the same host. HTTPS protocol is used for JOSSO

This scenario will show you how to secure your JBoss AS instance with JOSSO server and how to authenticate to JOSSO in secure way with usage of HTTPS protocol. Access to JOSSO gateway is secured and access to SOAP web services, which are used for communication between JOSSO gateway and JOSSO agent, is secured too. The assumption is that you did the previous scenario *2.1* and now you have JOSSO and JBoss Portal on the same host.

1. You have to enable HTTPS in your *default* configuration, where is JBoss Portal and JOSSO deployed. You should go through steps 1, 2, 3 of *CAS guide - section 1.2*.

2. Now only think you have to change is securing access to gateway and web services in file *$JBOSS_HOME/server/default/conf/josso-agent-config.xml*:

```
<!-- Gateway LOGIN and LOGOUT URLs -->
<gatewayLoginUrl>https://localhost:8443/josso/signon/login.do</gatewayLoginUrl>
<gatewayLogoutUrl>https://localhost:8443/josso/signon/logout.do</gatewayLogoutUrl>

<!-- Gateway service locator -->
<gatewayServiceLocator>
    <!-- Other properties for ws-service-locator :
     username, password, servicesWebContext, transportSecurity
   -->
   <protocol:ws-service-locator endpoint="localhost:8443" transportSecurity="confidential" />
</gatewayServiceLocator>
```

3. Now you should restart your JBoss AS, but you have to run it with special option -*Djavax.net.ssl.trustStore*. This Java option is really important because you are saying where is file with certificates, which should be trusted by Java. This is needed for correct usage of secured communication between web services.

   The whole command for starting JBoss should look like this:

```
./run.sh -Djavax.net.ssl.trustStore=$JBOSS_HOME/server/default/conf/server.keystore
```

4. When server is started, you can go to *http://localhost:8080/portal* and try to click to Login link. You shoud be redirected to JOSSO login screen and you can see that HTTPS is used for accessing JOSSO gateway. After successfull login, you should be redirected to JBoss Portal default page on *http://localhost:8080/portal/auth/portal/default* and you should see sign *Logged in as: admin* or *Logged in as: user* in right up corner.

   You can also use HTTPS for accessing JBoss Portal. HTTPS can be used on both sides of communication - on JBoss Portal side and on JOSSO side too.

2.3. Integration of JOSSO and more JBoss Portals deployed on multiple hosts

Next scenario simulates situation when two instances of JBoss Portal are deployed on different hosts and both are configured to authenticate against one JOSSO server. In this tutorial, we will simulate this scenario simply on one computer but we will use different configurations of JBoss AS (EAP) and we will use different host names for each configuration. In later text, the *josso-01* will be the name of the host and JBoss AS configuration where we will have JBoss Portal and JOSSO deployed. The *josso-02* will be the name of the host and JBoss AS configuration where we will have second JBoss Portal deployed. This second portal will be configured for authentication against JOSSO on josso-01.

1. Create new virtual hosts on your system and use some different names for them. In this scenario, We willl use names *josso-01* and *josso-02*. On linux systems, it can be donne by add these two lines

```
127.0.1.2       josso-01
127.0.1.3       josso-02
```

   to your */etc/hosts* file. You must have root privileges to do this. If you don't have root privileges, than you have to use IP addresses instead of host names in rest of this tutorial (For example: Running of JBoss Portal can be performed by *./run.sh -c josso-01 -b 127.0.1.2* instead of *./run.sh -c josso-01 -b josso-01*. You must use IP addresses in your configuration files too.).

2. Create JBoss AS configuration called *josso-01* and deploy JBoss Portal and JOSSO to it. We assume that you did previous scenario *2.2* and you have JBoss Portal with JOSSO properly configured in your default configuration. So only think you have to do is copy your *default* configuration to new *josso-01* configuration. This can be donne with command

```
cp -r $JBOSS_HOME/server/default $JBOSS_HOME/server/josso-01
```

```
```

on linux systems.

3. Create JBoss AS configuration called *josso-02* and deploy JBoss Portal to it. It's good to start *josso-02* (before installing of JOSSO agent) first without *-Dorg.apache.commons.logging.LogFactory* option to properly initialization of CMS. Then you can stop it. And then we will configure this portal to authenticate against JOSSO on *josso-01* host.

4. We need to deploy JOSSO agent to *josso-02* configuration. You can use *josso-gsh* in similar way, described in *2.1 - point 4*. Only change is, that you will deploy only agent but not gateway. You can install JOSSO agent with command:

```
agent install --target $JBOSS_HOME --jboss-instance josso-02 --platform jb42
```

5. You should configure your JBoss Portal on *josso-02* to authentication with JOSSO server on *josso-01*. Do steps 6, 8, 9, 10 of *2.1* scenario in josso-02 configuration.

6. Configure JOSSO agent on *josso-02* against JOSSO gateway on *josso-01*. Edit *$JBOSS_HOME/server/josso-02/conf/josso-agent-config.xml* and edit it for usage with JBoss Portal application. You should have this content inside your *<agent:partner-apps>* element.

```
<agent:partner-apps>
  <agent:partner-app id="jboss_portal" context="/portal"/>
</agent:partner-apps>
```

And gateway should be configured this way:

```
<!-- Gateway LOGIN and LOGOUT URLs -->
<gatewayLoginUrl>https://josso-01:8443/josso/signon/login.do</gatewayLoginUrl>
<gatewayLogoutUrl>https://josso-01:8443/josso/signon/logout.do</gatewayLogoutUrl>

<!-- Gateway service locator -->
<gatewayServiceLocator>
  <!-- Other properties for ws-service-locator :
   username, password, servicesWebContext, transportSecurity
  -->
  <protocol:ws-service-locator endpoint="josso-01:8443" transportSecurity="confidential" />
</gatewayServiceLocator>
```

7. You should generate new self-signed certificate for host *josso-01* because previous certificate was CN attribute with value *localhost*. Now we will use host *josso-01* for this configuration, so we have to generate new certificate with CN value josso-01.

   So go to directory *$JBOSS_HOME/server/josso-01/conf* and delete file *server.keystore* if one is here. Now you should generate new certificate with usage of command

```
keytool -genkey -alias jbosskey -keypass changeit -keyalg RSA -keystore server.keystore
```

   Answer *changeit* as value for keystore password and answer *josso-01* as answer for question about your first and second name. Other answers are not so important.

8. For secure communication between Web services on josso-01 side (gateway) and josso-02 (agent) you need to enable SSL in *josso-02* too. You should go through steps 1, 2, 3 of *CAS guide - section 1.2* but you have to use *josso-02* instead of default configuration and you should use CN name *josso-02* during certificate generation with keytool.

9. Configure JOSSO agent on *josso-01* against JOSSO gateway on *josso-01*. Edit *$JBOSS_HOME/server/josso-01/conf/josso-agent-config.xml* and edit it properly:

```
<!-- Gateway LOGIN and LOGOUT URLs -->
<gatewayLoginUrl>https://josso-01:8443/josso/signon/login.do</gatewayLoginUrl>
<gatewayLogoutUrl>https://josso-01:8443/josso/signon/logout.do</gatewayLogoutUrl>

<!-- Gateway service locator -->
<gatewayServiceLocator>
  <!-- Other properties for ws-service-locator :
   username, password, servicesWebContext, transportSecurity
  -->
  <protocol:ws-service-locator endpoint="josso-01:8443" transportSecurity="confidential" />
</gatewayServiceLocator>
```

10. Run *josso-01* configuration and bind it to *josso-01* host.

```
./run.sh -c josso-01 -b josso-01 -Djavax.net.ssl.trustStore=$JBOSS_HOME/server/josso-01/conf/server.keystore
```

11. Run *josso-02* configuration and bind it to *josso-02* host. You have to trust certificate from *josso-01* host.

```
./run.sh -c josso-02 -b josso-02 -Djavax.net.ssl.trustStore=$JBOSS_HOME/server/josso-01/conf/server.keystore
```

12. Test SSO. Go to *http://josso-01:8080/portal* and click to login link. You should be redirected to JOSSO login page on *https://josso-01:8443/josso/signon/login.do?josso_back_to=http://josso-01:8080/portal/josso_security_check*. Then you can check *Remember me* checkbox. After login and validating of your credentials, you should be logged and after that you are redirected back to portal on *http://josso-01:8080/portal/auth/portal/default*. Now try second JBoss Portal on *josso-02* host. Go to *http://josso-02:8080/portal* and click to login link. You are redirected directly to *http://josso-02:8080/portal/auth/portal/default* without need of filling any credentials. You can see that you are logged on *josso-02* with same user as on *josso-01*.

2.4. Integration of JOSSO, JBoss Portal and thirdparty web application

You can use JOSSO to authentication among many different web applications on different hosts. Only important condition is that you have these applications configured against same JOSSO server and you have properly configured them.

In next scenario we will deploy sample application and we will check that after login to this application, you are logged to JBoss Portal too. JOSSO came with sample application called *Partner application*. We will use it in this scenario. Go through these steps (assuming that you did previous scenario *2.3*):

1. Install sample partner JOSSO application. You can use *josso-gsh* in similar way, described in *2.1 - point 4*. You can install JOSSO partner application with command:

```
samples install --target $JBOSS_HOME --jboss-instance josso-01 --platform jb42
```

2. Enable your applications in *$JBOSS_HOME/server/josso-01/conf/josso-agent-config.xml* file:

```
<agent:partner-apps>
  <agent:partner-app id="jboss_portal" context="/portal"/>
  <agent:partner-app id="MyPartnerApp1" context="/partnerapp" />
</agent:partner-apps>
```

3. Test SSO. Go to *http://josso-01:8080/partnerapp* and click to login link. You should be redirected to JOSSO login page on *https://josso-01:8443/josso/signon/login.do?josso_back_to=http://josso-01:8080/partnerapp/josso_security_check*. Then you can check *Remember me* checkbox. After login and validating of your credentials (you should use same credentials as for login to JBoss Portal), you should be logged and after that you are

redirected back to partner application and you can see that you are logged in. You can see that you are in same roles as portal user with same name and you should be in *Authenticated* role as well.

Now try JBoss Portal. Go to *http://josso-01:8080/portal* and click to login link. You are redirected directly to *http://josso-01:8080/portal/auth/portal/default* without need of filling any credentials. You can see that you are logged on *josso-01* with same user as in partner application.

2.5. Integration of one JBoss Portal and one JOSSO server deployed on different hosts

The last scenario describes situation, when we want to deploy JBoss Portal in a host *portal-host* and we want to deploy JOSSO server to different host *josso-host*. Configuration of this scenario is slightly different especially the configuration of JOSSO gateway. We need to configure JOSSO gateway to authenticate directly against same Database, which is used by JBoss Portal.

In this scenario we will configure JBoss Portal with database and we will configure JOSSO to access directly the same database. JOSSO comes with set of standard identity stores and we will use one of them to bind Mysql database.

1. Create new virtual hosts on your system and use some different names for them. In this scenario, We will use names *portal-host* and *josso-host*. On linux systems, it can be donne by add these two lines

```
127.0.1.6       portal-host
127.0.1.8       josso-host
```

   to your */etc/hosts* file. You must have root privileges to do this. If you don't have root privileges, than you have to use IP addresses instead of host names in rest of this tutorial. In later text, josso-host will be a host where JOSSO server will be deployed (JBoss configuration will be *josso-conf*) and portal-host will be the host where JBoss Portal will be deployed (*default* configuration).

2. Take a fresh JBoss AS or JBoss EAP (JBoss-4.2.3.GA is good). Then create new configuration called *josso-conf* where JOSSO will be deployed. You can use this command in linux:

```
cp -r $JBOSS_HOME/server/default $JBOSS_HOME/server/josso-conf
```

3. Deploy JBoss Portal 2.7.2 to the default configuration of JBoss AS. We assume that you use Mysql database on localhost. So your datasource could look like this:

```
<datasources>
  <local-tx-datasource>
    <jndi-name>PortalDS</jndi-name>
          <connection-url>jdbc:mysql://localhost/jboss_portal?jdbcCompliantTruncation=false</
connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>jboss_portal</user-name>
    <password>password</password>
  </local-tx-datasource>
</datasources>
```

   You can use any other database than Mysql. Only important think is, that you must configure JOSSO against the same database as JBoss Portal in later steps of this scenario. You can read *Chapter 2* [http://docs.jboss.com/jbportal/v2.7.1/referenceGuide/html/installation.html] of portal reference guide if you have some troubles with deploying JBoss Portal configured against custom datasource.

4. Install JOSSO gateway to *josso-conf* and JOSSO agent to *default* configuration. Agent should be in same configuration where JBoss Portal is. Run this in josso-gsh:

```
gateway install --target $JBOSS_HOME --jboss-instance josso-conf --platform jb42
agent install --target $JBOSS_HOME --jboss-instance default --platform jb42
```

5. Do steps 6, 8, 9, 10 of *Section 2.1, “Integration of JBoss Portal and JOSSO deployed on the same host with usage of HTTP protocol”* in default configuration to Jossify JBoss Portal.

6. Configure JOSSO agent on *default* against JOSSO gateway on *josso-conf*. Edit *$JBOSS_HOME/server/default/conf/josso-agent-config.xml* and edit it for usage with JBoss Portal application. You should have this content inside your *<agent:partner-apps>* element.

```
<agent:partner-apps>
  <agent:partner-app id="jboss_portal" context="/portal"/>
</agent:partner-apps>
```

   And gateway should be configured this way:

```
<!-- Gateway LOGIN and LOGOUT URLs -->
<gatewayLoginUrl>http://josso-host:8080/josso/signon/login.do</gatewayLoginUrl>
<gatewayLogoutUrl>http://josso-host:8080/josso/signon/logout.do</gatewayLogoutUrl>

<!-- Gateway service locator -->
<gatewayServiceLocator>
  <!-- Other properties for ws-service-locator :
   username, password, servicesWebContext, transportSecurity
  -->
  <protocol:ws-service-locator endpoint="josso-host:8080" />
</gatewayServiceLocator>
```

7. Now we need to configure JOSSO gateway properly. First edit *$JBOSS_HOME/server/josso-conf/conf/josso-gateway-config.xml* and uncomment josso-db-stores.xml in Stores section:

```
<!-- Identity, Session and Assertion Stores configuration -->

<!--<s:import resource="josso-gateway-memory-stores.xml" />-->
<s:import resource="josso-gateway-db-stores.xml" />
<!--<s:import resource="josso-gateway-ldap-stores.xml" />-->
```

   In Authenticator section, you should uncomment basic authentication and others could be commented out:

```
<!--
 ========================================================================= -
->
<!-- SSO Authenticator, all authentication schemes must be configured here -->
<!--
 ========================================================================= -
->
<def-auth:authenticator id="josso-authenticator">
 <def-auth:schemes>
    <!--<s:ref bean="josso-bind-authentication"/>-->

    <s:ref bean="josso-basic-authentication"/>
    <!--<s:ref bean="josso-strong-authentication"/>
    <s:ref bean="josso-rememberme-authentication"/>
    -->
    <!-- Others like NTLM and BIND go here -->
```

```
 </def-auth:schemes>
</def-auth:authenticator>
```

8. Configure JOSSO gateway to authenticate against Database. You should edit *$JBOSS_HOME/server/josso-conf/conf/josso-gateway-db-stores.xml*. Comment *db-istore:datasource-store* section and uncomment and properly configure *db-istore:jdbc-store*. You need to configure database options and couple of SQL queries which are used by josso-db-identity-store:

```
<!--<db-istore:datasource-store id="josso-identity-store"
    dsJndiName="java:/DefaultDS"
    userQueryString="SELECT NAME FROM JOSSO_USER WHERE LOGIN = ?;"
    rolesQueryString="SELECT ROLE FROM JOSSO_USER_ROLE WHERE LOGIN = ?;"
          credentialsQueryString="SELECT LOGIN AS USERNAME, PASSWORD FROM
 JOSSO_USER WHERE LOGIN = ?;"
    userPropertiesQueryString="SELECT NAME, VALUE FROM JOSSO_USER_PROPERTY
 WHERE LOGIN = ?;"
   resetCredentialDml="UPDATE JOSSO_USER SET PASSWORD = ? WHERE LOGIN = ?;"
    relayCredentialQueryString="SELECT LOGIN FROM JOSSO_USER WHERE #?# = ?;"
/>-->

<db-istore:jdbc-store
          id="josso-identity-store"
          driverName="com.mysql.jdbc.Driver"
          connectionURL="jdbc:mysql://localhost:3306/jboss_portal"
          connectionName="jboss_portal"
          connectionPassword="password"
       userQueryString="SELECT jbp_uname AS NAME FROM jbp_users WHERE jbp_uname
 = ?;"
            rolesQueryString="SELECT jbp_roles.jbp_name AS ROLE FROM jbp_roles INNER
 JOIN jbp_role_membership AS jrm ON jrm.jbp_rid = jbp_roles.jbp_rid INNER JOIN jbp_users
 AS users ON users.jbp_uid = jrm.jbp_uid WHERE users.jbp_uname = ? UNION SELECT
 'Authenticated' AS ROLE;"
             credentialsQueryString="SELECT jbp_uname AS USERNAME, jbp_password AS
 PASSWORD FROM jbp_users WHERE jbp_uname = ?;"
      userPropertiesQueryString="SELECT jbp_name AS NAME, jbp_value AS VALUE FROM
 jbp_user_prop AS props INNER JOIN jbp_users AS users ON users.jbp_uid = props.jbp_uid
 WHERE jbp_uname = ?;"
         resetCredentialDml="UPDATE jbp_users SET jbp_password = ? WHERE jbp_uname
 = ?;"
       relayCredentialQueryString="SELECT jbp_uname AS LOGIN FROM jbp_users WHERE
 #?# = ?;"
```

```
/>
```

9. Configure *$JBOSS_HOME/server/josso-conf/conf/josso-gateway-auth.xml*. Uncomment and edit basic authentication:

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- Basic Authentication Scheme                               -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<basic-authscheme:basic-auth-scheme
      id="josso-basic-authentication"
      hashAlgorithm="MD5"
      hashEncoding="HEX"
      ignorePasswordCase="false"
      ignoreUserCase="false">

       <basic-authscheme:credentialStore>
          <s:ref bean="josso-identity-store"/>
       </basic-authscheme:credentialStore>

       <basic-authscheme:credentialStoreKeyAdapter>
          <s:ref bean="josso-simple-key-adapter"/>
       </basic-authscheme:credentialStoreKeyAdapter>

   </basic-authscheme:basic-auth-scheme>
```

Bind authentication should be commented out:

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- BIND Authentication Scheme (normally LDAP) -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- Requires a be a bindalble credential store ! -->
<!-- name attribute is important and must not be changed -->
<!--<bind-authscheme:bind-auth-scheme
        id="josso-bind-authentication"
        name="basic-authentication"
        hashAlgorithm="MD5"
        hashEncoding="HEX"
        ignorePasswordCase="false"
        ignoreUserCase="false">


     <bind-authscheme:credentialStore>
```

```
        <s:ref bean="josso-identity-store"/>
    </bind-authscheme:credentialStore>

    <bind-authscheme:credentialStoreKeyAdapter>
        <s:ref bean="josso-simple-key-adapter"/>
    </bind-authscheme:credentialStoreKeyAdapter>
</bind-authscheme:bind-auth-scheme>-->
```

10. Don't forget to add Mysql JDBC driver to *$JBOSS_HOME/server/josso-conf/deploy/josso.war/WEB-INF/lib* directory.

11. Start both configurations of JBoss AS and bind them to different hosts. First the default configuration with JBoss Portal:

```
./run.sh -c default -b portal-host
```

and then *josso-conf* configuration with JOSSO:

```
./run.sh -c josso-conf -b josso-host
```

12. Test integration. Go to *http://portal-host:8080/portal* and try to login. You should be redirected to JOSSO server *http://josso-host:8080/josso/signon/login.do?josso_back_to=http://portal-host:8080/portal/josso_security_check* and you should be able to login with default portal credentials.

2.6. FAQ

First take a look to *CAS FAQ* in previous chapter. Many scenarios are similar to JOSSO.

Which versions of JBoss Portal and JOSSO should I use?

All scenarios are tested with JBoss Portal 2.7.2 branch version (*http://anonsvn.jboss.org/repos/portal/branches/JBoss_Portal_Branch_2_7* [http://anonsvn.jboss.org/repos/portal/branches/JBoss_Portal_Branch_2_7/]) deployed on JBoss AS 4.2.3.GA. Version of JOSSO is 1.8.0. Configurations is different if you use different versions of components. For JOSSO 1.7.0 the configuration is very different and JBoss Portal 2.7.1 integrates only with JOSSO 1.7.0.

I have JBoss Portal configured against LDAP and I want to use JOSSO server for authentication deployed on different host than Portal

You can use LDAP identity store instead of Database identity store described in *2.5*. Take a look to *http://www.josso.org/confluence/display/JOSSO1/josso-ldap-identitystore* for more informations.

I have JBoss Portal integrated with JOSSO. But JOSSO server is unreachable right now

That's bad because when you attempt to login, then you are redirected to JOSSO login screen. And if this screen is unreachable you end with 404 error. So you are not able to login if JOSSO server is out.

How the JBoss Portal behaves when session timeout occurs?

When session timeout occurs in JBoss Portal, then session is invalidated. But after click to some link in JBoss Portal, the user is directly logged in without need to fill any credentials. The reason is, that if JOSSO gateway have a cookie with your credentials, then gateway informs JOSSO agent about it and the system logges you directly in. You are not forced to fill credentials again. So you are directly logged even if session attributes (and all portlet session attributes from portlet scope or application scope) are deleted.