JBoss Identity Federation

User Guide

1.0.0.alpha2.

by Anil Saldhana

What this Book Covers

This book aims to help you become familiar with JBoss Identity Federation in order that you can use it to build your own Federated Identity based services or applications.

Part I 'Getting Started' introduces the federated identity technologies that are provided in this product. It also indicates the libraries required for the installation.

Part II 'Simple Usage' describes SAML v2 Web Browser based Single Sign On (SSO).

Part III 'Advanced Usage' describes SAML v2 Web Browser based SSO with advanced features such as Trust Management and XML Digital Signatures.

Part IV 'Trouble Shooting' describes some basic troubleshooting tips when things do not work the way they were intended.

Part V 'Resources' provides additional resources.

Part I. Getting Started

Introduction

JBoss Identity Federation allows you to implement SAML v2.0 based services and applications. It also has support for Oasis WS-Trust based applications (which is under development).

With JBoss Identity Federation, you have the following features.

- SAML v2 Web Browser SSO (HTTP/Redirect Binding) Support for JBoss Application Server and Apache Tomcat.
- SAML v2 Web Browser SSO (HTTP/Redirect Binding) Support for JBoss Application Server and Apache Tomcat with XML Signature Support.

Installation

JBoss Identity Federation requires the following libraries to be either downloaded separately or as part of the Java JDK or as part of JBoss Application Server.

Download the ZIP version of the JBoss Identity Community Platform. Place the unzipped jar files in the lib directory of tomcat or JBoss AS. Additionally ensure that the following dependencies are met.

- JAXB V2 Library
- STAX API Library (a dependency for JAXB2)
- Activation API Library (a dependency for JAXB2)

Location for downloading the jars

JBoss Maven Repository [http://repository.jboss.org/maven2]

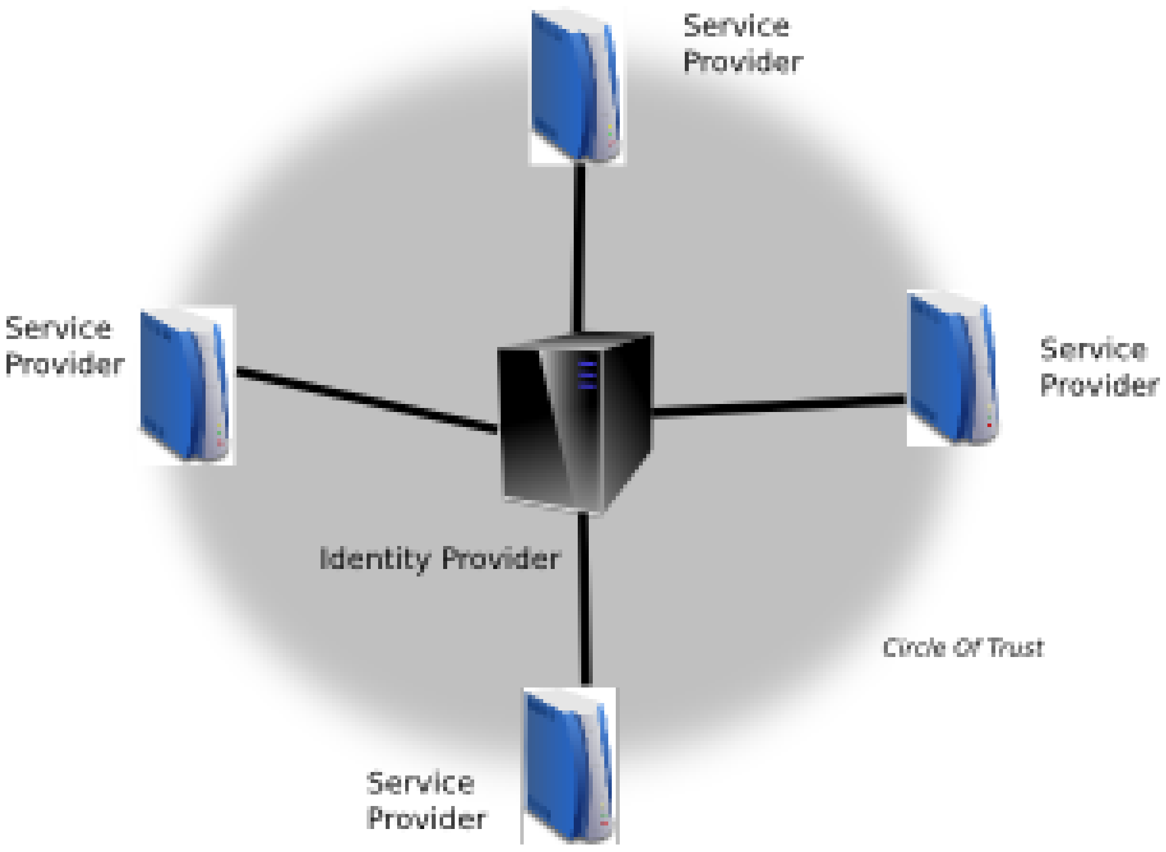
Part II. Simple Usage

Web Single Sign On (SSO)

In this chapter, we will look at usage of JBoss Identity Federation to help you obtain a platform to implement federated identity based services (including centralized identity services and Single Sign-On (SSO) for applications).

3.1. SAML v2 based Web SSO

This section will talk about the configuration information to support the SAML V2.0 based Web Single Sign On (SSO). The SAML profile that is implemented is the HTTP/Redirect binding with centralized identity services to enable web SSO for your applications.



Hub and Spoke Architecture for the SAML v2 based Web SSO

The architecture follows the Hub and Spoke architecture of Identity Management. An Identity Provider (IDP) acts as the central source (hub) for identity and role information to all the applications (Service Providers/SP). The spokes are the Service Providers (SP).

Note

The IDP and the SP can be a JBoss Application Server or a Tomcat instance. Please note that the instructions for Tomcat and JBAS are different.

3.1.1. Configuring the Identity Provider (IDP)

Check list for configuring the IDP

1. Configure the IDP as a secure web application.
2. Configure the web.xml to either allow FORM or BASIC authentication.
3. Configure the context.xml for IDP valves.
4. Configure the jboss-idfed.xml for IDP configuration.

The IDP can be a JBoss Application Server or a Tomcat instance.

You need to configure a web application as the Identity provider.

3.1.1.1. Configure the web application security for the IDP

The web application needs to have FORM or BASIC based security enabled in its web.xml. We recommend the use of FORM based web application security as it gives you the ability to customize the login page.

The web.xml needs to have a configuration such as the following:

```

        <?xml version="1.0" encoding="ISO-8859-1"?>
<web-app 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"
   version="2.5">

  <display-name>IDP</display-name>
  <description>IDP</description>

  <!-- Define a security constraint that gives unlimited access to images -->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Images</web-resource-name>
      <url-pattern>/images/*</url-pattern>
    </web-resource-collection>
  </security-constraint>

    <!-- Define a Security Constraint on this Application -->
```

```
 <security-constraint>
   <web-resource-collection>
     <web-resource-name>IDP</web-resource-name>
     <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>manager</role-name>
   </auth-constraint>
 </security-constraint>

 <!-- Define the Login Configuration for this Application -->
 <login-config>
   <auth-method>FORM</auth-method>
   <realm-name>IDP Application</realm-name>
   <form-login-config>
      <form-login-page>/jsp/login.jsp</form-login-page>
      <form-error-page>/jsp/loginerror.jsp</form-error-page>
   </form-login-config>
 </login-config>

 <!-- Security roles referenced by this web application -->
 <security-role>
   <description>
     The role that is required to log in to the IDP Application
   </description>
   <role-name>manager</role-name>
 </security-role>
</web-app>
```

Note

Remember to configure the realm or login modules for your IDP as per the Tomcat or JBoss AS documentation on "securing your web application". *Tomcat Realm* [http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html] and *JBoss AS Security* [http://jboss.org/community/docs/DOC-10760]

3.1.1.2. Configure the IDP Valves

Create a *context.xml* file for configuring the valves for the IDP.

The context.xml file should look like:

```
<Context>
 <Valve
    className="org.jboss.identity.federation.bindings.tomcat.idp.IDPRedirectValve"
/>
</Context>
```

Note

If the IDP is running in Apache Tomcat, then place the context.xml in **META-INF** of your IDP web application.

Note

If the IDP is running in JBoss Application Server, then place the context.xml in **WEB-INF** of your IDP web application.

3.1.1.3. Configure the JBoss Identity Federation configuration file (jboss-idfed.xml)

Configure *jboss-idfed.xml* in WEB-INF of your IDP web application

```
<JBossIDP xmlns="urn:jboss:identity-federation:config:1.0" >
  <IdentityURL>http://localhost:8080/idp</IdentityURL>
</JBossIDP>
```

In this configuration file, you are providing the URL of your IDP. This is the URL that gets added as the issuer in the outgoing SAML2 assertions to the Service Providers.

3.1.2. Configure the Service Provider (SP)

Check List for configuring the Service Provider.

1. Configure the SP as a secure FORM authentication based web application.
2. Configure the web.xml of the SP web application.

3. Configure the context.xml for the SP valves.
4. Configure the jboss-idfed.xml for the SP configuration.
5. Perform additional steps if the SP is running on JBoss Application Server.

The SP can be a JBoss Application Server or a Tomcat instance.

You need to configure a web application as the Service Provider(SP).

3.1.2.1. Configure the web application security for the SP

The web application needs to have FORM based security enabled in its web.xml.

The web.xml needs to have a configuration such as the following:

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
<web-app 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"
   version="2.5">

  <display-name>Test SALES Application</display-name>
  <description>
    Just a Test SP
  </description>

  <!-- Define a Security Constraint on this Application -->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>SALES Application</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
       <role-name>manager</role-name>
    </auth-constraint>
  </security-constraint>

  <!-- Define a security constraint that gives unlimted access to freezone -->
  <security-constraint>
    <web-resource-collection>
     <web-resource-name>freezone</web-resource-name>
     <url-pattern>/freezone/*</url-pattern>
    </web-resource-collection>
  </security-constraint>

  <!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Tomcat SALES Application</realm-name>
    <form-login-config>
       <form-login-page>/jsp/login.jsp</form-login-page>
       <form-error-page>/jsp/loginerror.jsp</form-error-page>
    </form-login-config>
  </login-config>

  <!-- Security roles referenced by this web application -->
  <security-role>
    <description>
      The role that is required to log in to the SP Application
    </description>
    <role-name>manager</role-name>
  </security-role>
</web-app>
```

Warning

The SP web application should be configured with FORM based authentication.

3.1.2.2. Configure the SP Valves

Create a *context.xml* file for configuring the valves for the SP.

The context.xml file should look like:

```
            <Context>
              <Valve

 className="org.jboss.identity.federation.bindings.tomcat.sp.SPRedirectFormAuthenticator"
              />
            </Context>
```

Note

If the SP is running in Apache Tomcat, then place the context.xml in **META-INF** of your SP web application.

Note

If the SP is running in JBoss Application Server, then place the context.xml in **WEB-INF** of your SP web application.

3.1.2.3. Configure the JBoss Identity Federation configuration file (jboss-idfed.xml)

Configure *jboss-idfed.xml* in WEB-INF of your SP web application

```
<JBossSP xmlns="urn:jboss:identity-federation:config:1.0">
  <IdentityURL>http://localhost:8080/idp</IdentityURL>
  <ServiceURL>http://localhost:8080/sales</ServiceURL>
</JBossSP>
```

In this configuration file, we define the URLs for the service provider and the identity provider.

3.1.2.4. Additional Steps for JBoss AS based SP

Configure a *jboss-web.xml* file in the WEB-INF directory of your SP web application.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss-web
    PUBLIC "-//JBoss//DTD Web Application 2.4//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-web_4_0.dtd">
<jboss-web>
   <security-domain>java:/jaas/sp</security-domain>
```

```
</jboss-web>
```

Note

In this example, we have specified a security domain of "sp". You can use any security domain name of your choice as long as you configure the login module in the next step appropriately.

Configure the login module in conf/login-config.xml of your JBoss AS server configuration.

```
<application-policy name = "sp">
  <authentication>
    <login-module
      code = "org.jboss.identity.federation.bindings.jboss.auth.SAML2LoginModule" />
  </authentication>
</application-policy>
```

Part III. Advanced Usage (Trust Management)

Web SSO - XML Signature Support

In this chapter, we describe the configuration for Web SSO with XML Signature Support.

4.1. Configuring the Identity Provider

The IDP needs to be configured to provide Web SSO with XML Signature Support.

Check list for configuring the IDP

1. Configure the IDP as a secure web application.
2. Configure the web.xml to either allow FORM or BASIC authentication.
3. Configure the context.xml for IDP valves.
4. Configure the jboss-idfed.xml for IDP configuration.

4.1.1. Configure the IDP Web Application Security

Configure the web application security for IDP

Follow the web.xml security configuration for the IDP from the previous section "Simple Usage".

4.1.2. Configure the IDP Valves

Create a *context.xml* file for configuring the valves for the IDP.

The context.xml file should look like:

```
<Context>
 <Valve
   className
   ="org.jboss.identity.federation.bindings.tomcat.idp.IDPRedirectWithSignatureValve"
/>
</Context>
```

Note

If the IDP is running in Apache Tomcat, then place the context.xml in **META-INF** of your IDP web application.

Note

If the IDP is running in JBoss Application Server, then place the context.xml in **WEB-INF** of your IDP web application.

4.1.3. Configure the JBoss Identity Federation configuration file (jboss-idfed.xml)

Configure *jboss-idfed.xml* in WEB-INF of your IDP web application

```
<JBossIDP xmlns="urn:jboss:identity-federation:config:1.0" >
  <IdentityURL>http://localhost:8080/idp-sig</IdentityURL>
  <Trust>
     <Domains>localhost,jboss.com,jboss.org</Domains>
  </Trust>
  <KeyProvider
     ClassName="org.jboss.identity.federation.bindings.tomcat.KeyStoreKeyManager">
     <Auth Key="KeyStoreURL" Value="jbid_test_keystore.jks" />
     <Auth Key="KeyStorePass" Value="store123" />
     <Auth Key="SigningKeyPass" Value="test123" />
     <Auth Key="SigningKeyAlias" Value="servercert" />
     <ValidatingAlias Key="localhost" Value="servercert"/>
     <ValidatingAlias Key="127.0.0.1" Value="servercert"/>
  </KeyProvider>
</JBossIDP>
```

In this configuration file, you are providing the URL of your IDP. This is the URL that gets added as the issuer in the outgoing SAML2 assertions to the Service Providers.

Additionally, you can configure the *Trust* element to indicate which domains the IDP trusts.

You can configure a **TrustKeyManager** implementation for the Signing (Private) Key and the Validating (Public) Key information. In this example, we have used the **KeyStoreKeyManager** that stores the keys in a Java KeyStore. The *Auth* element define the key value pair needed to authenticate against the **TrustKeyManager** implementation. The *ValidatingAlias* is a map of the domains that need to be validated against an alias where the public key of the domains are stored.

4.2. Configure the Service Provider (SP)

Check List for configuring the Service Provider.

1. Configure the SP as a secure FORM authentication based web application.
2. Configure the web.xml of the SP web application.
3. Configure the context.xml for the SP valves.
4. Configure the jboss-idfed.xml for the SP configuration.

The SP can be a JBoss Application Server or a Tomcat instance.

You need to configure a web application as the Service Provider(SP).

4.2.1. Configure the SP Web Application Security

Configure the web application security for SP

Follow the web.xml security configuration for the SP from the previous section "Simple Usage".

4.2.2. Configure the SP Valves

Create a *context.xml* file for configuring the valves for the SP.

The context.xml file should look like:

```
<Context>
  <Valve
  className=
"org.jboss.identity.federation.bindings.tomcat.sp.SPRedirectSignatureFormAuthenticator"
  />
</Context>
```

Note

If the SP is running in Apache Tomcat, then place the context.xml in **META-INF** of your SP web application.

Note

If the SP is running in JBoss Application Server, then place the context.xml in **WEB-INF** of your SP web application.

4.2.3. Configure the JBoss Identity Federation configuration file (jboss-idfed.xml)

Configure *jboss-idfed.xml* in WEB-INF of your IDP web application

```
<JBossIDP xmlns="urn:jboss:identity-federation:config:1.0" >
  <IdentityURL>http://localhost:8080/idp-sig</IdentityURL>
  <Trust>
     <Domains>localhost,jboss.com,jboss.org</Domains>
  </Trust>
  <KeyProvider
     ClassName="org.jboss.identity.federation.bindings.tomcat.KeyStoreKeyManager">
     <Auth Key="KeyStoreURL" Value="jbid_test_keystore.jks" />
     <Auth Key="KeyStorePass" Value="store123" />
     <Auth Key="SigningKeyPass" Value="test123" />
     <Auth Key="SigningKeyAlias" Value="servercert" />
     <ValidatingAlias Key="localhost" Value="servercert"/>
     <ValidatingAlias Key="127.0.0.1" Value="servercert"/>
  </KeyProvider>
</JBossIDP>
```

In this configuration file, we define the URLs for the service provider and the identity provider.

Additionally, you can configure the *Trust* element to indicate which domains the SP trusts.

You can configure a **TrustKeyManager** implementation for the Signing (Private) Key and the Validating (Public) Key information. In this example, we have used the **KeyStoreKeyManager** that stores the keys in a Java KeyStore. The *Auth* element define the key value pair needed to authenticate against the **TrustKeyManager** implementation. The *ValidatingAlias* is a map of the domains that need to be validated against an alias where the public key of the domains are stored.

5.1. Web SSO (XML Encryption Support)

Part IV. Troubleshooting

6.1. Configuring Logging

JBoss Identity Federation uses Apache log4j as the logging framework.

6.1.1. Configuring Logging on Apache Tomcat

Log4J jars and xml file

Add a log4j.jar (from the Apache log4j Distribution) into the lib directory of tomcat 6.x or server/lib of tomcat 5.5.x

Also add a log4j.xml as shown below to the lib directory.

```
          <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<!-- =====================================================================
 -->
<!--                                                -->
<!--  Log4j Configuration                                   -->
<!--                                                -->
<!-- =====================================================================
 -->

<!--
  | For more configuration information and examples see the Jakarta Log4j
  | owebsite: http://jakarta.apache.org/log4j
 -->

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/" debug="false">

   <!-- ================================= -->
   <!-- Preserve messages in a local file -->
   <!-- ================================= -->

   <!-- A time/date based rolling appender -->
   <appender name="FILE" class="org.apache.log4j.DailyRollingFileAppender">
      <param name="File" value="../logs/server.log"/>
      <param name="Append" value="false"/>
      <!--
         Set the threshold via a system property. Note this is parsed by log4j,
      so the full JBoss system property format is not supported; e.g.
      setting a default via ${jboss.server.log.threshold:WARN} will not work.
     -->
    <param name="Threshold" value="TRACE"/>

    <!-- Rollover at midnight each day -->
    <param name="DatePattern" value="'.'yyyy-MM-dd"/>

    <!-- Rollover at the top of each hour
    <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>
    -->

    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date Priority [Category] (Thread) Message\n -->
      <param name="ConversionPattern" value="%d %-5p [%c] (%t) %m%n"/>

      <!-- The full pattern: Date MS Priority [Category] (Thread:NDC) Message\n
      <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%t:%x) %m%n"/>
       -->
    </layout>
  </appender>

  <!-- ============================== -->
  <!-- Append messages to the console -->
  <!-- ============================== -->

  <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.out"/>
    <param name="Threshold" value="INFO"/>

    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date Priority [Category] Message\n -->
      <param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c{1}] %m%n"/>
    </layout>
  </appender>

  <!-- ================ -->
  <!-- Limit categories -->
  <!-- ================ -->

  <!-- Limit the org.apache category to INFO as its DEBUG is verbose -->
  <category name="org.apache">
    <priority value="TRACE"/>
  </category>
```

```
   <category name="org.jboss">
      <priority value="TRACE"/>
   </category>

   <!-- Setup the Root category -->
   <!-- ======================= -->

   <root>
      <appender-ref ref="CONSOLE"/>
      <appender-ref ref="FILE"/>
   </root>

</log4j:configuration>
```

Location of the generated log file

The generated log file will be server.log in the logs directory.

6.1.2. Configuring logging in JBoss

You can configure log4j in the conf directory of your JBoss server (default, all etc)

Tip

Please refer to JBoss AS documentation on logging.

Part V. Resources

Resources on the Web

JBossIdentity Project Page [http://www.jboss.org/jbossidentity]

JBoss Identity User Forum [http://www.jboss.org/index.html?module=bb&op=viewforum&f=305]

JBoss Identity Design Forum [http://www.jboss.com/index.html?module=bb&c=32]