WsmSecurityModel

In WSM, there're currently three kinds of security models. They're "Servlet container security model", "Axis security model" and "Beehive security model".

This page describes their usages, advantages and disadvantages.

We will use Atm.jws (Automatic Teller Machine) below to explain each security model.

```
// Atm.jws
import javax.jws.Oneway;
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public class Atm {

    @WebMethod
    @SecurityRoles(rolesAllowed={"admin","customer"})
    public int withdraw(int amount)
    {
        return amount;
    }

    @WebMethod
    @SecurityRoles(rolesAllowed={"admin","customer"})
    public int getBalance()
    {
        return 1000;
    }

    @WebMethod
    @Oneway
    @SecurityRoles(rolesAllowed={"admin","engineer"})
    public void fix()
    {
        return;
    }

    @WebMethod
    @Oneway
    public void showStatus()
    {
        return;
    }
}
```

- admin role can access all methods.
- customer role can access the withdraw, getBalance and showStatus methods.
- engineer role can access the fix and showStatus method.
- No restrictions to access the showStatus method. (Everybody can access the method.)

To enable any of security models for WSM on Axis, you have to set up two steps: set up the role information and modify server-config.wsdd.

To set up role information.

Servlet container security model

With this model, you can reuse the existing roles of your servlet container.

Assume your web.xml has the following security-role elements.

```
<security-role>
 <role-name>admin</role-name>
</security-role>
<security-role>
 <role-name>customer</role-name>
</security-role>
<security-role>
 <role-name>engineer</role-name>
</security-role>
```

To apply those roles for Atm.jws, you add the following elements in web.xml.

```
<security-constraint>
 <web-resource-collection>
   <url-pattern>/Atm.jws</url-pattern>
 </web-resource-collection>
 <auth-constraint>
   <role-name>admin</role-name>
   <role-name>customer</role-name>
   <role-name>engineer</role-name>
 </auth-constraint>
</security-constraint>

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

The disadvantage of this model is that one must have any one of roles listed in auth-constraint element, even when accessing a non-restricted method. For example, the showStatus() method of Atm.jws is not restricted because of absence of @SecurityRoles annotation, but one must have at least one of admin, customer or engineer role to access the method. This happens because the servlet container denies the user to access the Atm.jws without roles before the user reaches the web service.

Axis security model

You can reuse usernames of Axis's users.lst using this model. NOTE: usernames in users.lst are treated as role names of the @SecurityRoles annotation.

Here is the example of users.lst

```
admin admin_pass
customer customer_pass
engineer engineer_pass
```

The disadvantage of this model is that this security model is a user-based authentication and uses a plain text for password.

Beehive security model

This model is ported from Tomcat memory realm (using tomcat-users.xml file). To use this model, you must create a file named beehive-role.xml and place it in WEB-INF directory of your web service application.

Here is the example beehive-role.xml.

```
<beehive-role xmlns="http://www.apache.org/beehive/wsm/axis/security/xmlbeans">
   <role name="admin">
      <user>michael</user>
   </role>
   <role name="engineer">
      <user>jonathan</user>
      <user>dims</user>
   </role>
   <role name="customer">
      <user>jonathan</user>
      <user>wolfgang</user>
   </role>
   <user name="michael" password="1f2dfa567dcf95833eddf7aec167fec7" md5="true" />
   <user name="jonathan" password="jp" />
   <user name="dims" password="dp" />
   <user name="wolfgang" password="wp" />
</beehive-role>
```

You can use a md5 digest(32bytes HEX) for your password instead a plain text with a md5 attribute set "true" in a user tag. Absence of the md5 attribute in a user tag is equivalent to md5="false" then your password should be in a plain text.

To get a md5 digest of your password, there's a md5sum command in linux box. e.x) % md5sum --string="your password"

NOTE: The default namespace (xmlns="http://www.apache.org/beehive/wsm/axis/security/xmlbeans") must be specified in the root beehive-role tag.

To set up server-config.wsdd

Here is the example server-config.wsdd for servlet container security model.

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment xmlns="http://xml.apache.org/axis/wsdd/" xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"
>
    <globalConfiguration>
        <parameter name="adminPassword" value="admin"/>
        <parameter name="disablePrettyXML" value="true"/>
        <parameter name="attachments.implementation" value="org.apache.axis.attachments.AttachmentsImpl"/>
        <parameter name="sendXsiTypes" value="true"/>
        <!--<parameter name="sendMultiRefs" value="true"/>-->
        <parameter name="sendXMLDeclaration" value="true"/>
        <requestFlow>

            <handler type="java:org.apache.beehive.wsm.axis.AnnotatedWebServiceDeploymentHandler">
                <parameter name="scope" value="session"/>
            </handler>

            <handler type="java:org.apache.beehive.wsm.axis.AuthenticationHandler">
                <parameter name="securityModel" value="org.apache.beehive.wsm.axis.security.model.
ServletSecurityModel"/>
            </handler>

            <handler type="java:org.apache.axis.handlers.JWSHandler">
                <parameter name="scope" value="session"/>
            </handler>

        </requestFlow>
    </globalConfiguration>
    <handler name="LocalResponder" type="java:org.apache.axis.transport.local.LocalResponder"/>
    <handler name="URLMapper" type="java:org.apache.axis.handlers.http.URLMapper"/>
    <handler name="Authenticate" type="java:org.apache.axis.handlers.SimpleAuthenticationHandler"/>
    <service name="AdminService" provider="java:MSG">
        <parameter name="allowedMethods" value="AdminService"/>
        <parameter name="enableRemoteAdmin" value="false"/>
        <parameter name="className" value="org.apache.axis.utils.Admin"/>
        <namespace>http://xml.apache.org/axis/wsdd/</namespace>
    </service>
    <service name="Version" provider="java:RPC">
        <parameter name="allowedMethods" value="getVersion"/>
        <parameter name="className" value="org.apache.axis.Version"/>
    </service>
    <transport name="http">
        <requestFlow>
            <handler type="URLMapper"/>
            <handler type="java:org.apache.axis.handlers.http.HTTPAuthHandler"/>
        </requestFlow>
        <parameter name="qs:list" value="org.apache.axis.transport.http.QSListHandler"/>
        <parameter name="qs:wsdl" value="org.apache.axis.transport.http.QSWSDLHandler"/>
        <parameter name="qs:method" value="org.apache.axis.transport.http.QSMethodHandler"/>
    </transport>
    <transport name="local">
        <responseFlow>
            <handler type="LocalResponder"/>
        </responseFlow>
    </transport>
</deployment>
```

You can find the handler element below in server-config.wsdd above. This is for servlet security model.

```
<handler type="java:org.apache.beehive.wsm.axis.AuthenticationHandler">
   <parameter name="securityModel" value="org.apache.beehive.wsm.axis.security.model.ServletSecurityModel"/>
</handler>
```

To use Axis security model, please replace the handler above with lines below.

```
<handler type="java:org.apache.beehive.wsm.axis.AuthenticationHandler">
   <parameter name="securityModel" value="org.apache.beehive.wsm.axis.security.model.AxisSecurityModel"/>
</handler>
```

This is for Beehive security model. To use this model, please replace the handler above with lines below.

```
<handler type="java:org.apache.beehive.wsm.axis.AuthenticationHandler">
   <parameter name="securityModel" value="org.apache.beehive.wsm.axis.security.model.BeehiveMemorySecurityModel"
/>
</handler>
```

TBD