Secure Web Services with Apache Rampart

By Ruchith Fernando
ruchithf@apache.org
ruchith@wso2.com

Introduction to Axis2

- Next generation SOAP processing framework
- AXIOM – StAX based XML infoset representation with MTOM support
- QoS as self contained modules

Introduction to WSS4J

- OASIS Web Services Security (WS-Security)
  - 1.0 and 1.1
- UsernameToken profile
- X509 Token profile
- WS-SecureConversation
- WS-Trust
- WS-SecurityPolicy

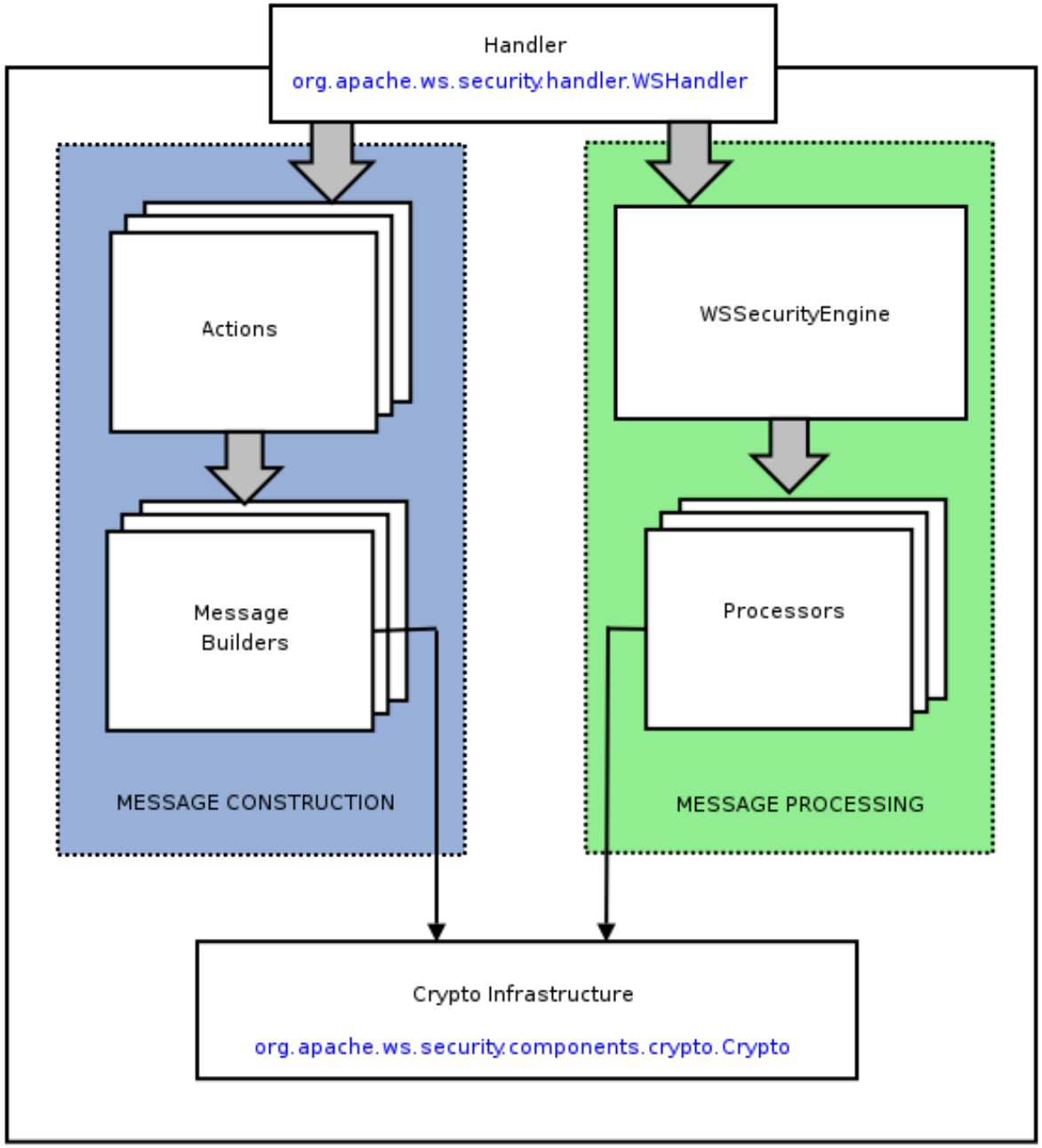
Introduction to WSS4J contd...

- Features
  - UsernameToken
  - Timestamp
  - Encryption
  - Signature

Supports

  - Axis 1.x
  - **Axis2 module – Rampart**

WSS4J Architecture


Handler
org.apache.ws.security.handler.WSHandler
Actions
Message Builders
MESSAGE CONSTRUCTION
WSSecurityEngine
Processors
MESSAGE PROCESSING
Crypto Infrastructure
org.apache.ws.security.components.crypto.Crypto

WSS4J Core

WSS4J &Axis2 -> Rampart

- WSS4J is depends on Apache XML-Security for Signature and Encryption
- XML Security is based on **DOM !!!**
- Axis2 is based on **OM/StAX !!!**

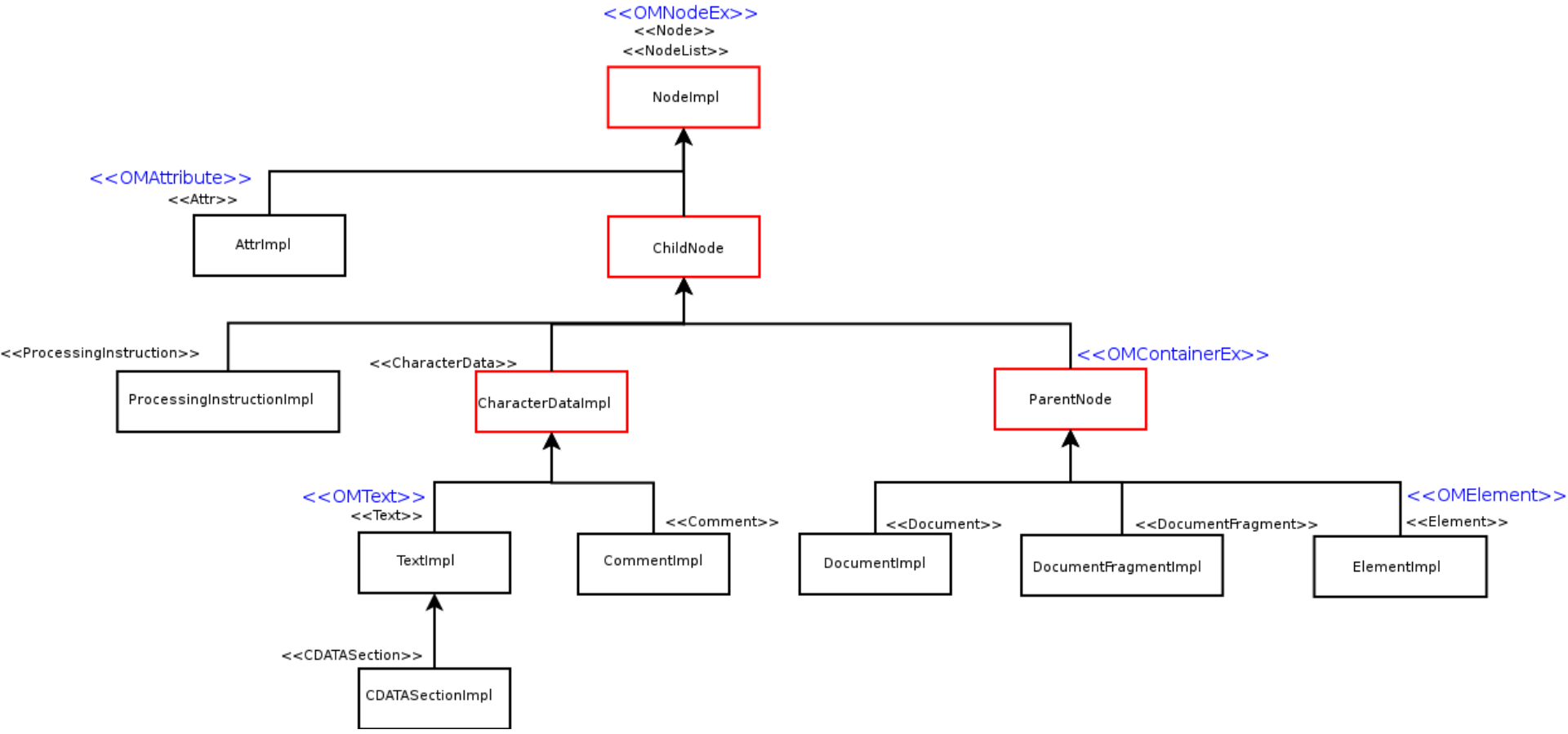
PROBLEM !!!

SOLUTION ??

DOM + OM = DOOM

- An implementation that supports both DOM and OM

DOOM


<<OMNodeEx>>
<<Node>>
<<NodeList>>
NodeImpl
<<OMAttribute>>
<<Attr>>
AttrImpl
ChildNode
<<ProcessingInstruction>>
ProcessingInstructionImpl
<<CharacterData>>
CharacterDataImpl
<<OMContainerEx>>
ParentNode
<<OMText>>
<<Text>>
TextImpl
<<Comment>>
CommentImpl
<<Document>>
DocumentImpl
<<DocumentFragment>>
DocumentFragmentImpl
<<OMElement>>
<<Element>>
ElementImpl
<<CDATASection>>
CDATASectionImpl

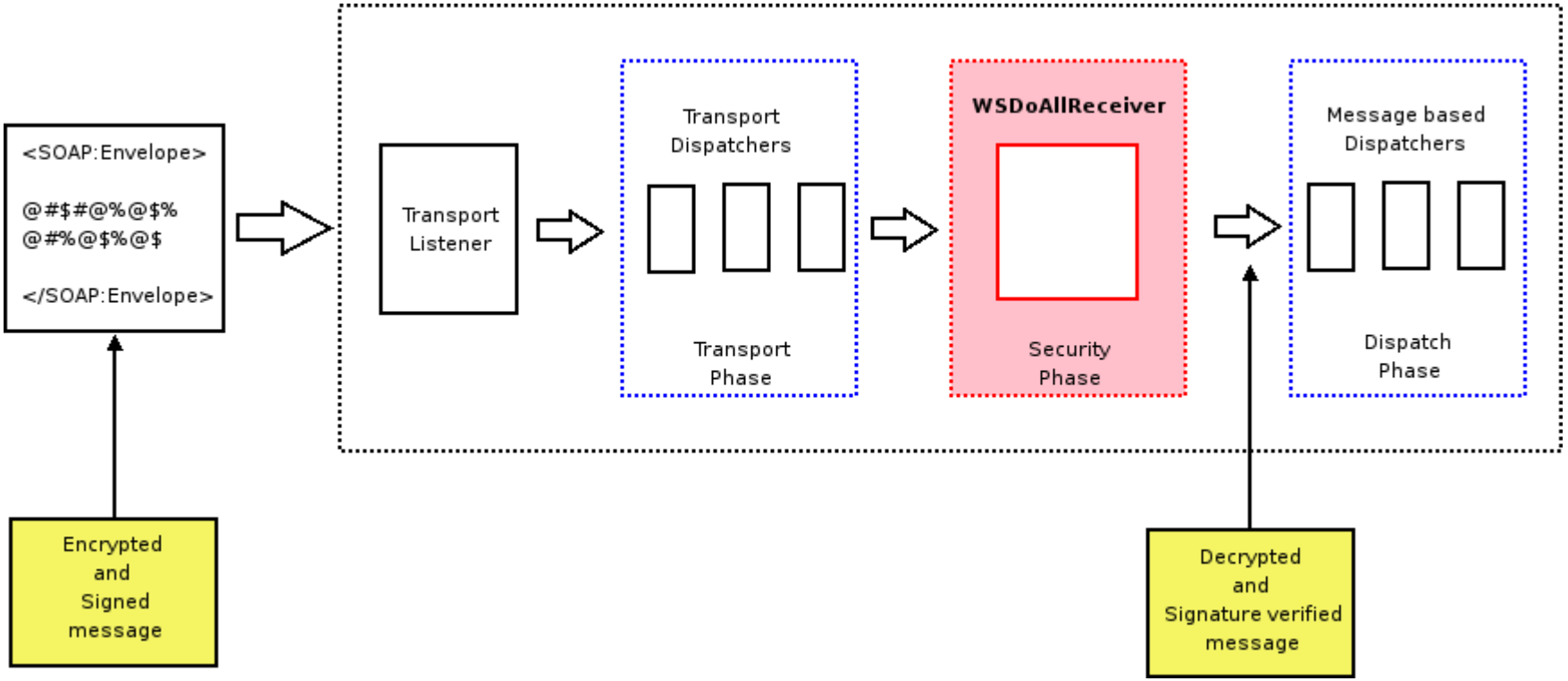
Rampart

Axis2 Module

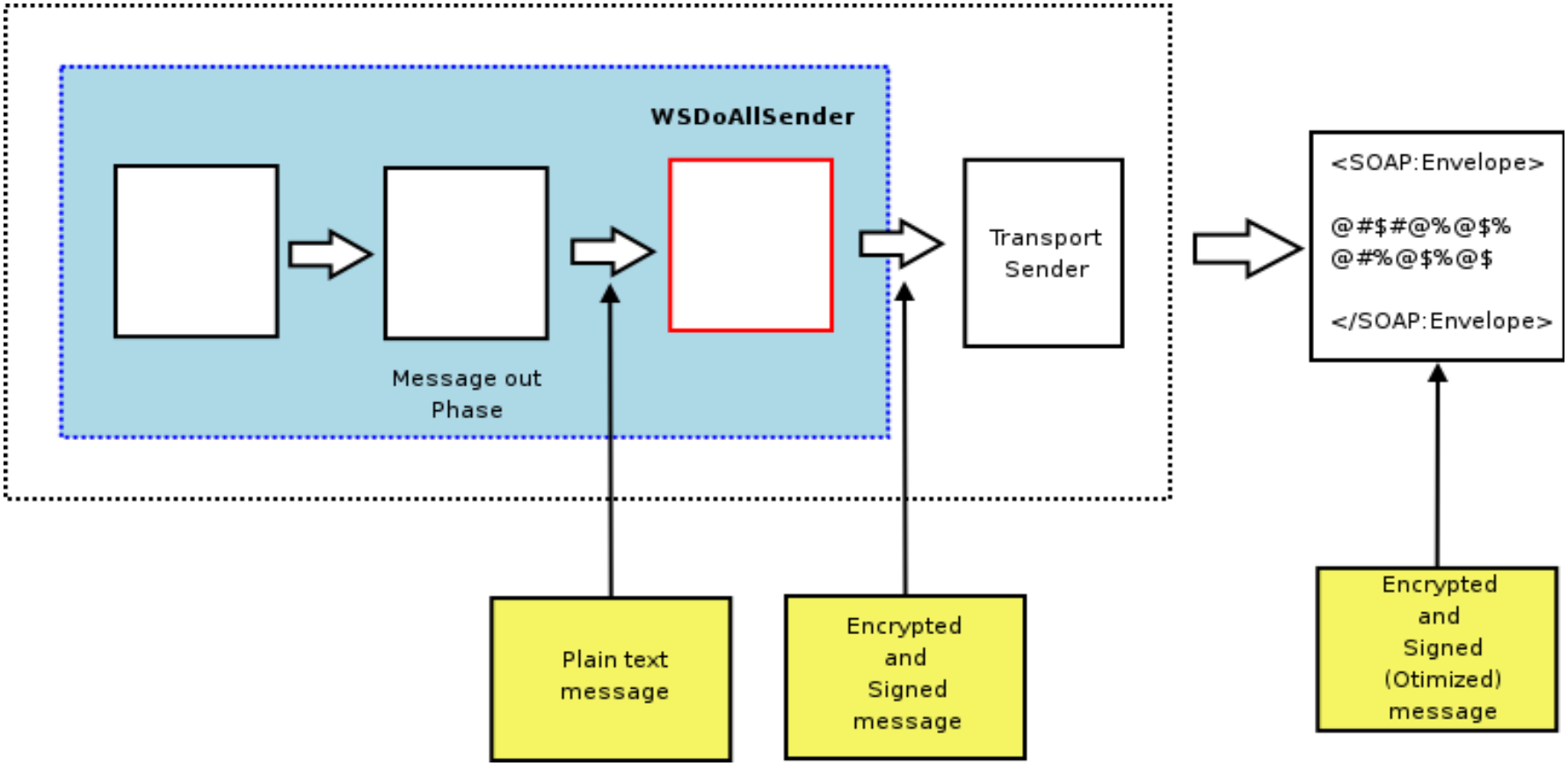
- Self contained
- Processing units -> handlers
- Automatically places them in the appropriate places in the flows

Rampart - Architecture

Inflow

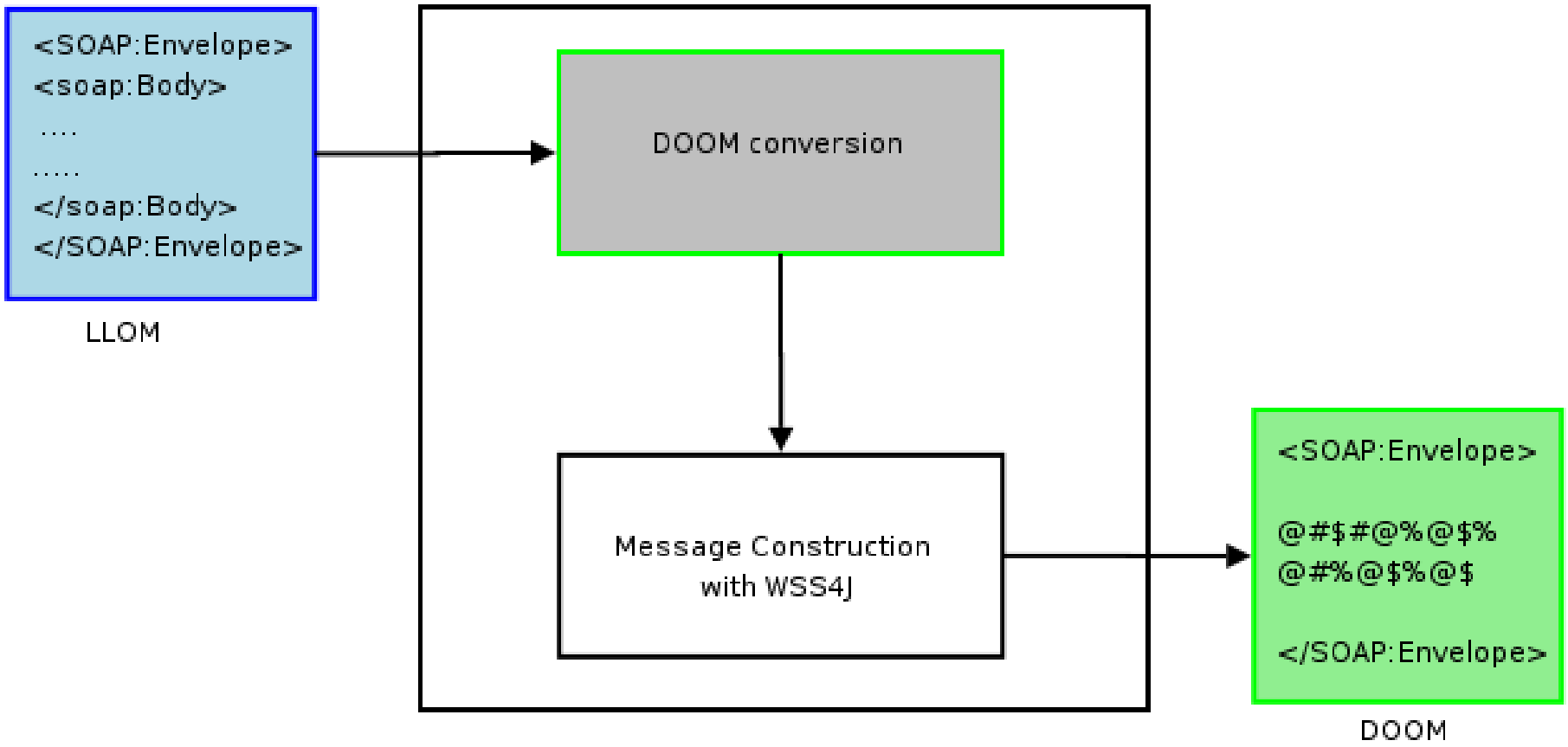

<SOAP:Envelope>
@#$#@%@$%
@#%@$%@$
</SOAP:Envelope>
Transport Listener
Transport Dispatchers
Transport Phase
WSDoAllReceiver
Security Phase
Message based Dispatchers
Dispatch Phase
Encrypted and Signed message
Decrypted and Signature verified message

Outflow

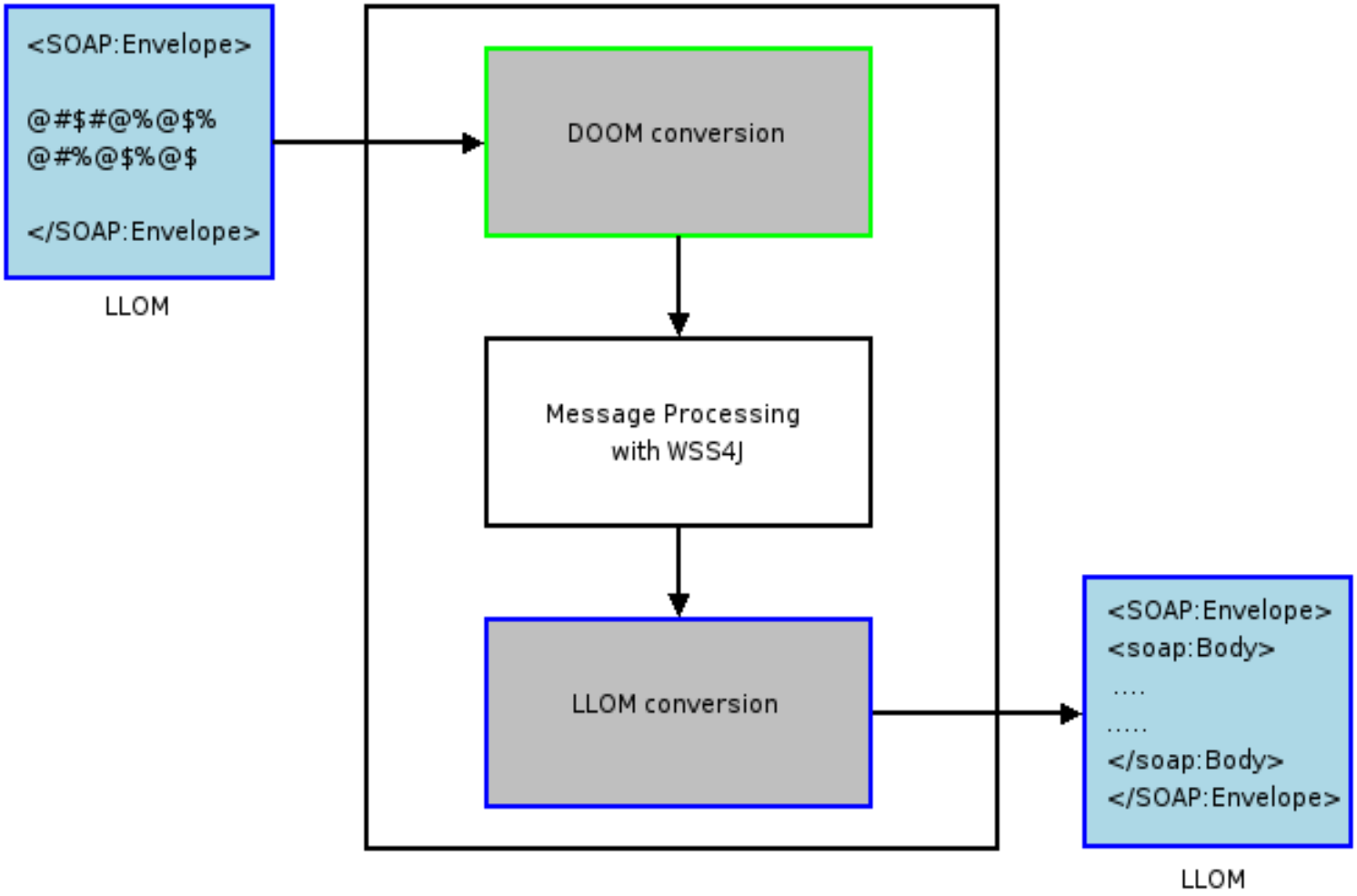

WSDoAllSender
Message out
Phase
Transport
Sender
<SOAP:Envelope>
@#$#@%@$%
@#%@$%@$
</SOAP:Envelope>
Plain text
message
Encrypted
and
Signed
message
Encrypted
and
Signed
(Otimized)
message

DOOM in Action !!!

Rampart Outflow Handler



Rampart Inflow Handler



Configuring Rampart

- Engaging rampart does not enforce any security on the messages
- Must explicitly configure rampart
- Two ways to configure
  - Static – Using descriptors
  - Dynamic – Using rampart configuration classes

Rampart Features

- Timestamp
- UsernameToken
- Signature
- Encryption
- MTOM optimization of base64 content
- Experimental WS-SecurityPolicy support

Lets get our client and service ready to try “rampart”

www-lk.wso2.com/~ruchith/rampart-tute.zip

Sample 01: No Security

Sample 01: No Security

- “rampart” engaged
- sample01/services.xml

```
<module ref=“rampart” />
```

- sample01/client.axis2.xml

```
<module ref=“rampart” />
```

Sample 01: No Security

- Host the service

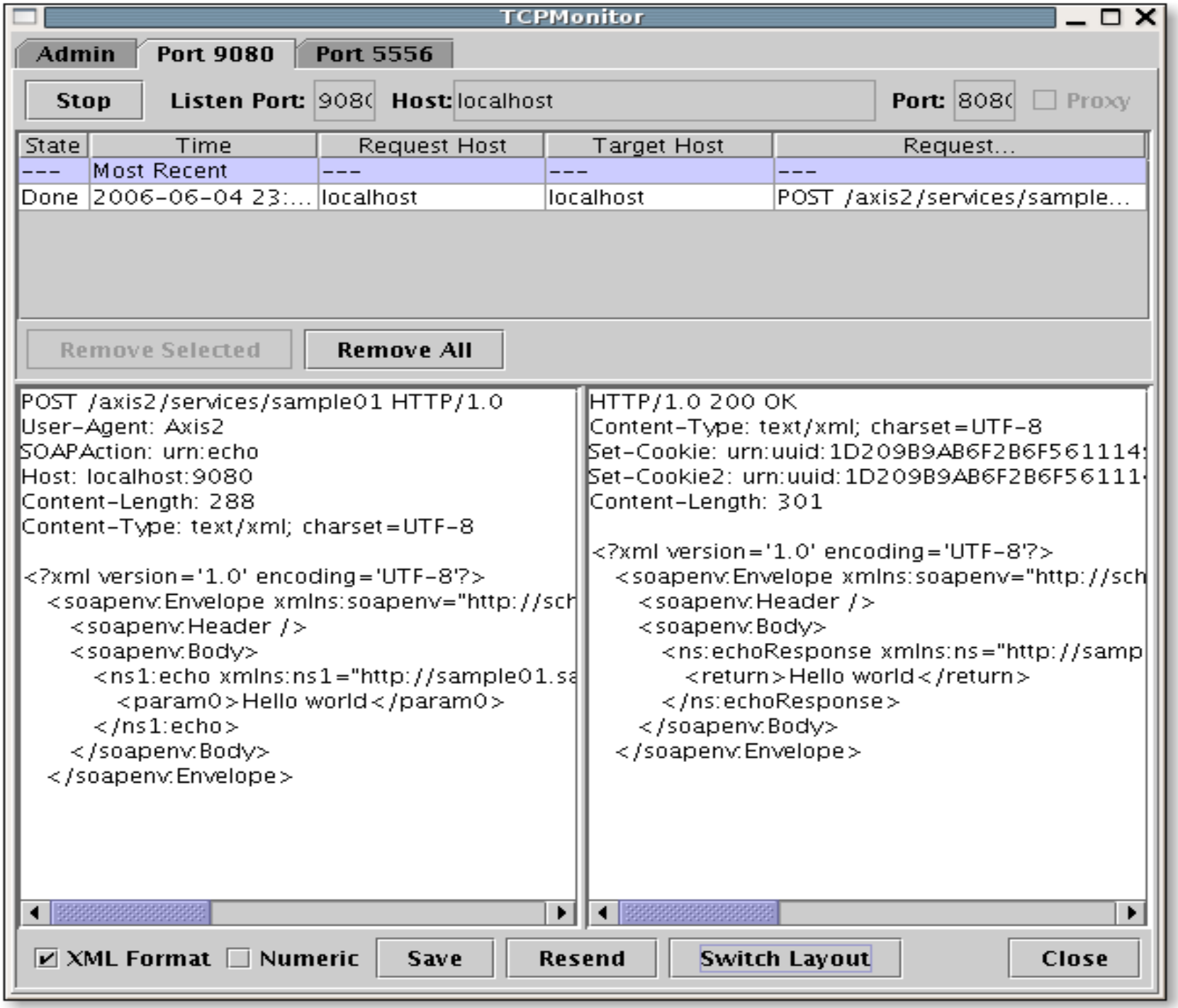
```
$ ant service.01
```

- Run client

```
$ ant client.01
```

Start the “tcpmon” to see the messages

- Change the “`client.port`” property value in the build.xml


TCPMonitor
Admin
Port 9080
Port 5556
Stop
Listen Port: 9080
Host: localhost
Port: 8080
Proxy
State
Time
Request Host
Target Host
Request...

Most Recent

Done
2006-06-04 23:...
localhost
localhost
POST /axis2/services/sample...
Remove Selected
Remove All
POST /axis2/services/sample01 HTTP/1.0
User-Agent: Axis2
SOAPAction: urn:echo
Host: localhost:9080
Content-Length: 288
Content-Type: text/xml; charset=UTF-8
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://sch
<soapenv:Header />
<soapenv:Body>
<ns1:echo xmlns:ns1="http://sample01.sa
<param0>Hello world</param0>
</ns1:echo>
</soapenv:Body>
</soapenv:Envelope>
HTTP/1.0 200 OK
Content-Type: text/xml; charset=UTF-8
Set-Cookie: urn:uuid:1D209B9AB6F2B6F561114
Set-Cookie2: urn:uuid:1D209B9AB6F2B6F56111
Content-Length: 301
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://sch
<soapenv:Header />
<soapenv:Body>
<ns:echoResponse xmlns:ns="http://samp
<return>Hello world</return>
</ns:echoResponse>
</soapenv:Body>
</soapenv:Envelope>
XML Format
Numeric
Save
Resend
Switch Layout
Close

Sample 02
UsernameToken Authentication

Client Configuration - axis2.xml

- sample02/client.axis2.xml

```
<parameter name="OutflowSecurity">
   <action>
       <items>UsernameToken</items>
   </action>
</parameter>
```

Client Configuration - axis2.xml

- sample02/client.axis2.xml

```
<parameter name="OutflowSecurity">
   <action>
       <items>UsernameToken</items>
     <user>bob</user>
   </action>
</parameter>
```

Password ?

passwordCallbackClass

- sample02/client.axis2.xml

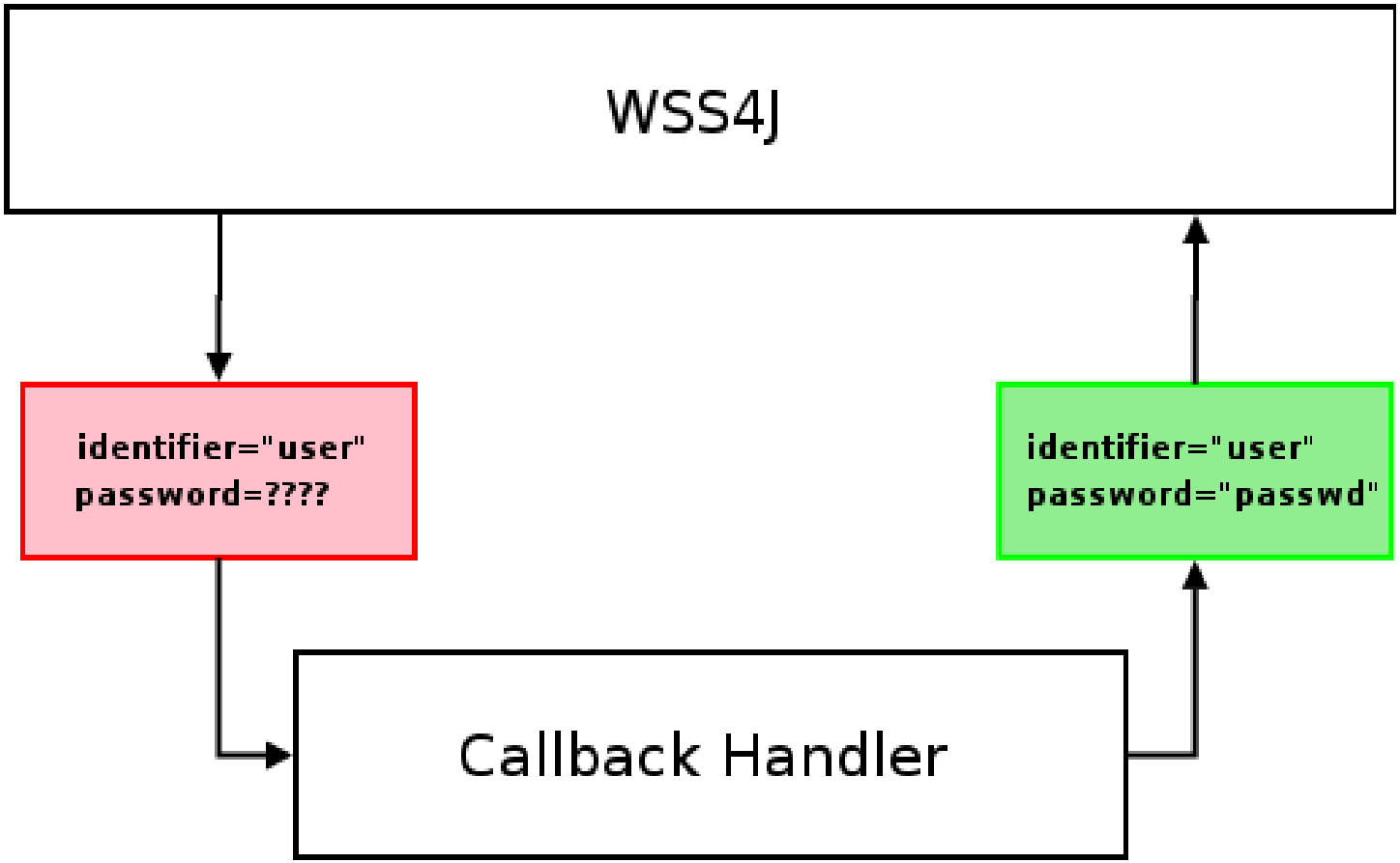
```
<parameter name="OutflowSecurity">
   <action>
     <items>UsernameToken</items>
     <user>bob</user>
     <passwordCallbackClass>CallbackHandler</passwordCallbackClass>
   </action>
</parameter>
```

passwordCallbackClass

- org.apache.rampart.samples.sample02.PWCB Handler
- org.apache.ws.security.WSPasswordCallback

```
public void handle(Callback[] callbacks) throws IOException,
        UnsupportedCallbackException {
    for (int i = 0; i < callbacks.length; i++) {
        WSPasswordCallback pwcb =
(WSPasswordCallback)callbacks[i];
        String id = pwcb.getIdentifer();
        if("bob".equals(id)) {
            pwcb.setPassword("bobPW");
        }
    }
```

Fetching the password



Service Configuration - services.xml

- sample02/services.xml

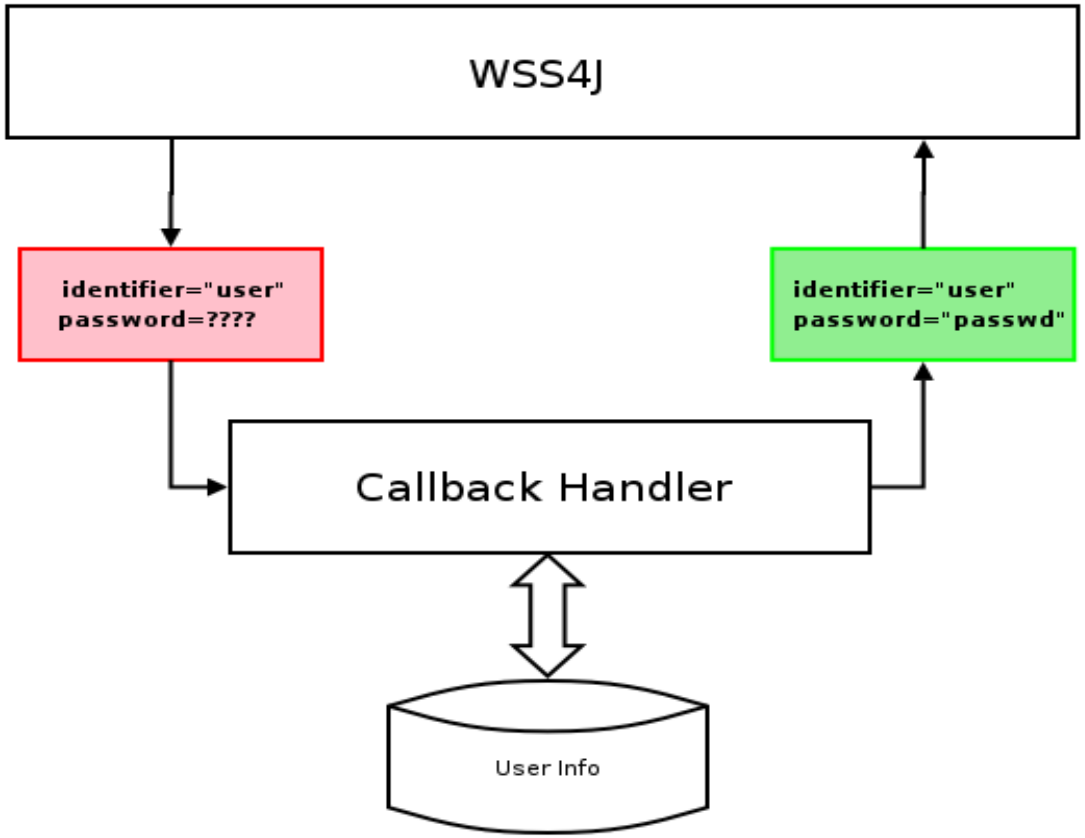
```
<parameter name="InflowSecurity">
    <action>
        <items>UsernameToken</items>
    </action>
</parameter>
```

Service Configuration - services.xml

- sample02/services.xml

```
<parameter name="InflowSecurity">
    <action>
        <items>UsernameToken</items>
    <passwordCallbackClass>PWCBHandler</passwordCallbackClass>
    </action>
</parameter>
```

Users can be fetched from any external store



Host sample 02 service and run the client

Sample 02 : Request

Notice the `wsse:UsernameToken` element in the `wsse:Security` header

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

  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

    <soapenv:Header>

      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
  200401-wss-wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">

        <wsse:UsernameToken xmlns:wsu="...oasis-200401-wss-wssecurity-utility-
  1.0.xsd" wsu:Id="UsernameToken-2745175">

          <wsse:Username>bob</wsse:Username>

          <wsse:Password Type="...#PasswordDigest">
  WyYODu0J2Msvf/QwS9dKKMxPawI=</wsse:Password>

          <wsse:Nonce>NyL/EBGcX2fRL2YvdYSnHQ==</wsse:Nonce>

          <wsu:Created>2006-06-05T05:02:23.399Z</wsu:Created>

        </wsse:UsernameToken>

      </wsse:Security>

    </soapenv:Header>

    <soapenv:Body>

    .....

    .....

    </soapenv:Body>
```

Password Types

- Two types of passwords
  - Password Digest
    - A digest of the actual password
    - pw = SHA-1(password+nonce+created)
  - Password Text
    - Password in plain text

Sample 03 : Plain Text Password

Sample 03: Client Configuration

- sample03/client.axis2.xml
- PasswordType is set to "`PasswordText`"
- Rampart/WSS4J doesn't authenticate the user.
- Handle our own user authentication at the callback handlers

How do we extract the user information at the service?

sample03/src/org/apache/rampart/samples/sample03/PWCBHandler.java

Integrity and Non-repudiation

Sample 04 :
Timestamp and Signature

Sample 04 : Client Configuration

- sample04/client.axis2.xml

```
<parameter name="OutflowSecurity">
    <action>
        <items>Timestamp Signature</items>
    </action>
</parameter>
```

Sample 04 : Client Configuration User

- sample04/client.axis2.xml

```
<parameter name="OutflowSecurity">
    <action>
        <items>Timestamp Signature</items>
        <user>client</user>
    </action>
</parameter>
```

Sample 04 : Client Configuration Crypto Information

- sample04/client.axis2.xml

```
<parameter name="OutflowSecurity">
   <action>
      <items>Timestamp Signature</items>
      <user>client</user>
  <signaturePropFile>client.properties</signaturePropFile>
   </action>
</parameter>
```

Crypto Information

- Implementation of `org.apache.ws.security.components.crypto.Crypto`
- Default implementation (Merlin) supports Java and PKCS12 keystores
- .properties file to configure WSS4J with the information

keys/client.properties

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.components.crypto.Merlin

org.apache.ws.security.crypto.merlin.keystore.type=jks

org.apache.ws.security.crypto.merlin.keystore.password=apache

org.apache.ws.security.crypto.merlin.file=client.jks
```

List contents of the keystore

```
$ keytool -list -v -keystore keys/client.jks -storepass apache
```

Sample 04 : Client Configuration
Password of the private key?

- sample04/client.axis2.xml

```
<parameter name="OutflowSecurity">
    <action>
        <items>Timestamp Signature</items>
        <user>client</user>
        <signaturePropFile>client.properties</signaturePropFile>
        <passwordCallbackClass>org.apache.rampart.samples.sample04.PWCBHandler</passwordCallbackClass>
    </action>
</parameter>
```

Sample 04 : Client Configuration
How to reference the signing key?

- sample04/client.axis2.xml

```
<parameter name="OutflowSecurity">
   <action>
      <items>Timestamp Signature</items>
      <user>client</user>
      <signaturePropFile>client.properties</signaturePropFile>
   <passwordCallbackClass>PWCBHandler</passwordCallbackClass>
<signatureKeyIdentifier>DirectReference</signatureKeyIdentifier>
   </action>
</parameter>
```

Sample 04 : Service Configuration

- sample04/services.xml

```
<parameter name="InflowSecurity">
   <action>
      <items>Timestamp Signature</items>
  <signaturePropFile>service.properties</
  signaturePropFile>
   </action>
</parameter>
```

Confidentiality

Sample 05 : Encryption

Sample 05 : Client Configuration

- sample05/client.axis2.xml

```
<parameter name="OutflowSecurity">
    <action>
      <items>Encrypt</items>
    </action>
</parameter>
```

Sample 05 : Client Configuration

- sample05/client.axis2.xml

```
<parameter name="OutflowSecurity">
    <action>
      <items>Encrypt</items>
<encryptionUser>service</encryptionUser>
    </action>
</parameter>
```

Sample 05 : Client Configuration

- sample05/client.axis2.xml

```
<parameter name="OutflowSecurity">
    <action>
      <items>Encrypt</items>
      <encryptionUser>service</encryptionUser>
<encryptionPropFile>client.properties</encryptio
  nPropFile>
    </action>
</parameter>
```

Sample 05 : Service Configuration

- sample05/services.xml

```
<parameter name="InflowSecurity">
    <action>
        <items>Encrypt</items>
    <passwordCallbackClass>org.apache.rampart.samples.
    sample05.PWCBHandler</passwordCallbackClass>
<decryptionPropFile>service.properties</decryptionPr
    opFile>
    </action>
</parameter>
```

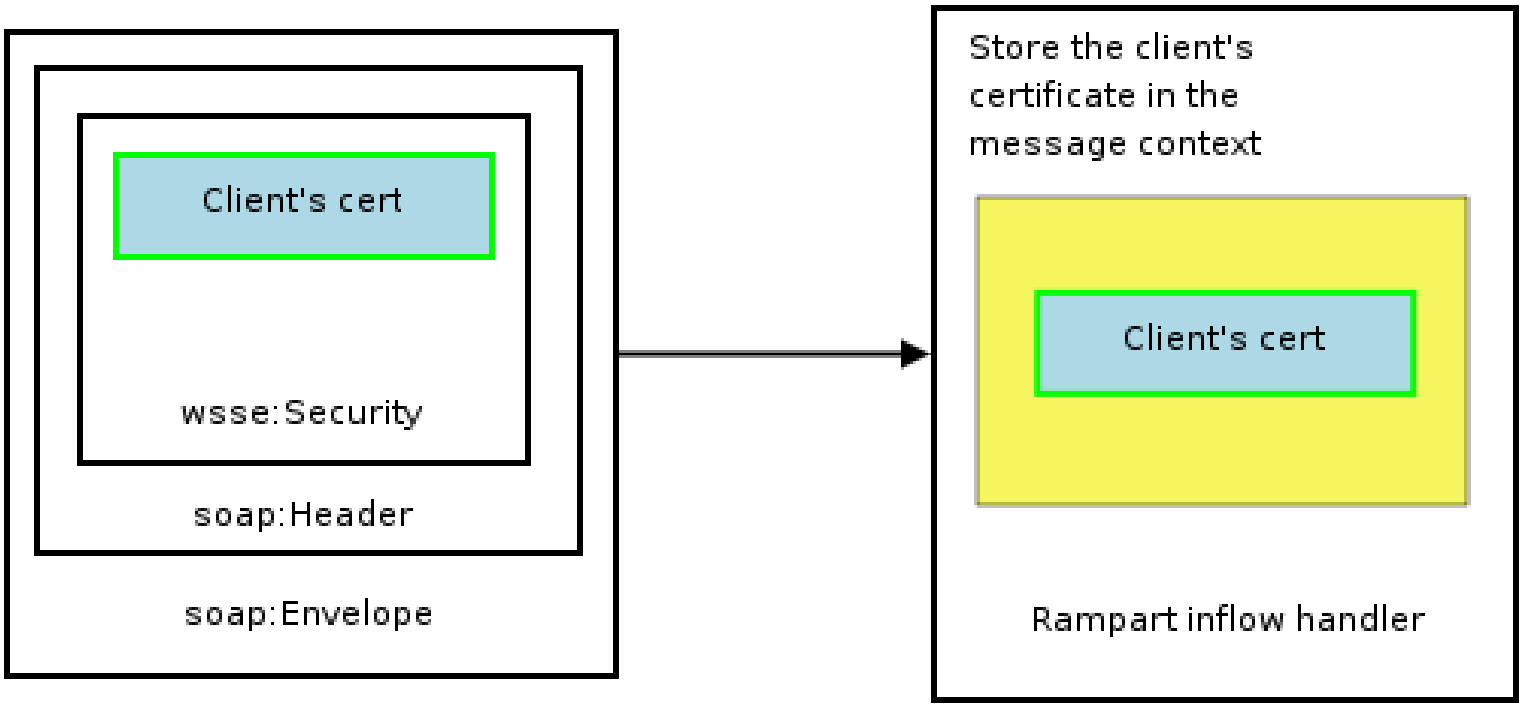
The response from sample 05 service is encrypted for the client

- Uses the client's certificate
- **`<encryptionUser>client</encryptionUser>`**
- How do we handle multiple clients?

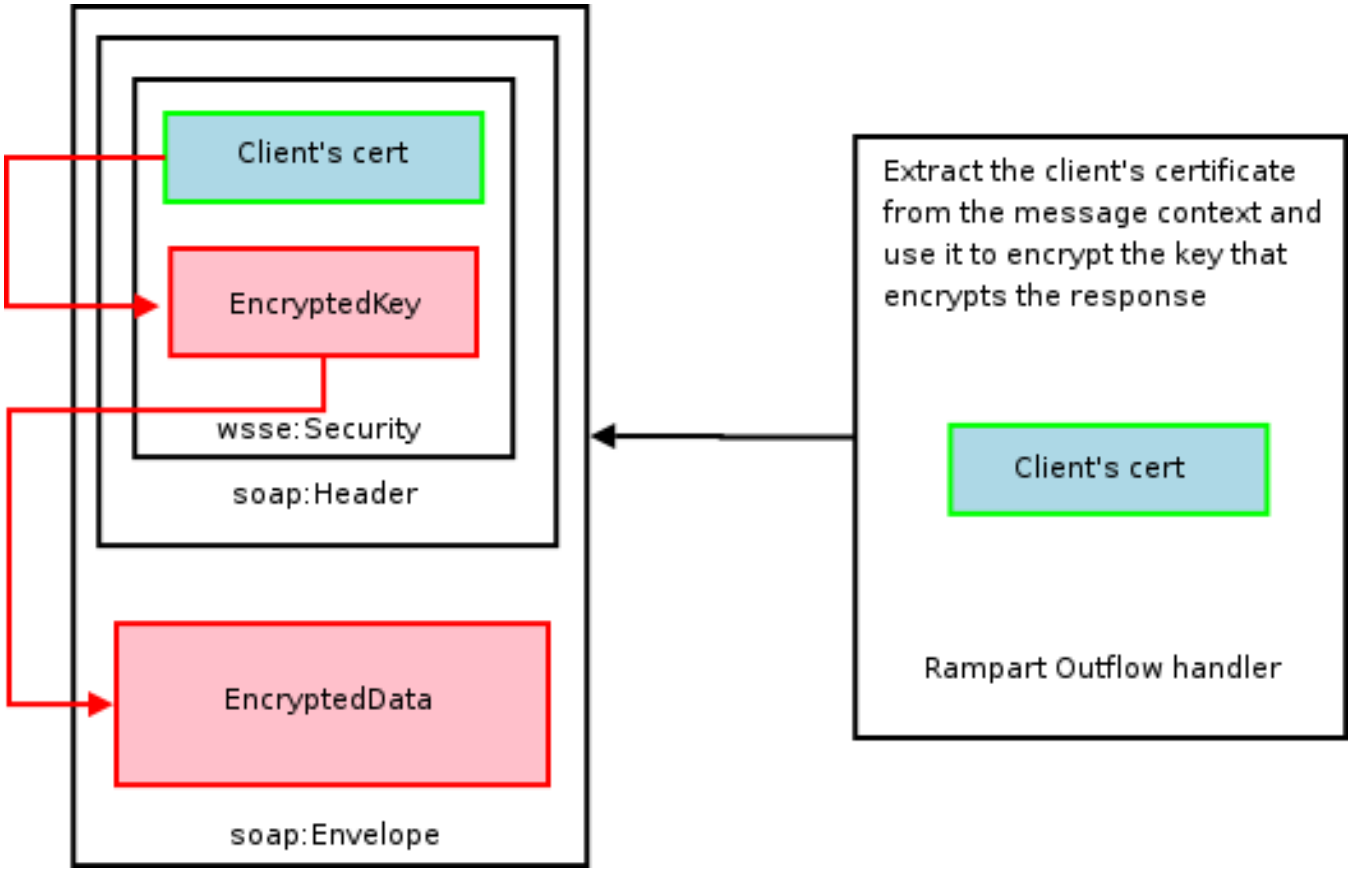
Handling Multiple Clients

- Service should be able to identify a client
- Use the certificate that was used to sign the request (if there is a Signature)
- **`<encryptionUser>useReqSigCert</encryptionUser>`**

Extract the Client Certificate



Use the Client Cert to Encrypt



Lets try it
Sample 06 :
Signature and Encryption

Sample 06 : Client Configuration

- sample06/client.axis2.xml
- Note the action/items

```
<parameter name="OutflowSecurity">
    <action>
        <items>Timestamp Signature Encrypt</items>
        ...
        ...
    </action>
</parameter>
```

Sample 06 : Service Configuration

- sample06/services.xml
- **signaturePropFile** used in absence of **decryptionPropFile**

```
<parameter name="InflowSecurity">
   <action>
      <items>Timestamp Signature Encrypt</items>
<passwordCallbackClass>...PWCBHandler</passwordCallbackCl
  ass>
<signaturePropFile>service.properties</signaturePropFile>
   </action>
</parameter>
```

We can sign and then encrypt

....

What about encrypt and sign ?

Sample 07 : Encrypt and Sign

- Simply change the order of the action/items in both client and service configurations

- **`<items>Encrypt Signature</items>`**

Sample 08 :
Signing Twice

Sample 08 : Client Configuration

- sample08/client.axis2.xml
- OutflowSecurity parameter with two `<action>` elements
- Two users can sign different parts of the message

Sample 08 : Client Configuration

```
<parameter name="OutflowSecurity">
   <action>
      <items>Timestamp Signature
  NoSerialization</items>
      <user>client</user>
   </action>
   <action>
      <items>Signature</items>
      <!-- This user can be someone else -->
      <user>client</user>
   </action>
</parameter>
```

Sample 08 : Service Configuration

- sample08/service.xml

```
<parameter name="InflowSecurity">
   <action>
   <items>Timestamp Signature Signature</items>
<passwordCallbackClass>...PWCBHandler</passwordCallb
  ackClass>
<signaturePropFile>service.properties</signatureProp
  File>
   </action>
</parameter>
```

When the client and service already has a shared key, can we use that to encrypt?

Sample 09 : Client Configuration

- sample09/client.axis2.xml

```
<parameter name="OutflowSecurity">

   <action>

<encryptionKeyIdentifier>EmbeddedKeyName</encryption
  KeyIdentifier>

<EmbeddedKeyCallbackClass>...PWCBHandler</EmbeddedKe
  yCallbackClass>

<EmbeddedKeyName>SessionKey</EmbeddedKeyName>

   ...

   ...

   </action>

</parameter>
```

Have a close look at the EmbeddedKeyCallbackClass

`org.apache.rampart.samples.sample09.PWCBHandler`

```
  if (pwcb.getUsage() ==
                WSPasswordCallback.KEY_NAME) {
    pwcb.setKey(key);
   }
```

Sample 09 : Service Configuration

- sample09/services.xml
-

```
<parameter name="InflowSecurity">

   <action>

       <items>Encrypt</items>
  <passwordCallbackClass>...PWCBHandler</passwordCallbackClass>
  <decryptionPropFile>service.properties</decryptionPropFile>

   </action>

</parameter>
```

OM supports MTOM

Axis2 Uses OM

Can we make use of this??

We can use MTOM optimization on Base64 text nodes

- XML-Encryption requires the xenc:EncryptedData/xenc:CipherData /xenc:CipherValue to be base64 encoded

- We can mark this text value as to be MTOM optimized

```
<?xml version='1.0' encoding='UTF-8'?><soapenv:Envelope >
<soapenv:Header>
<wsse:Security>
   ...
</wsse:Security>
</soapenv:Header>
<soapenv:Body>
   <xenc:EncryptedData>
      <xenc:CipherData>
         <xenc:CipherValue>QahoUagALkzOErllWyTaLpcVQDw==
         </xenc:CipherValue>
      </xenc:CipherData>
   </xenc:EncryptedData>
</soapenv:Body>
```

```
POST /axis2/services/sample10 HTTP/1.0.
User-Agent: Axis2.
SOAPAction: urn:echo.
Host: localhost:9080.
Content-Length: 5617.
Content-Type: multipart/related; boundary=MIMEBoundaryurn_uuid_D3D8A265C4834B582A11495537312962; type="application/xo
.
--MIMEBoundaryurn_uuid_D3D8A265C4834B582A11495537312962.
content-type:application/xop+xml; charset=UTF-8; type="text/xml";.
content-transfer-encoding:binary.
content-id:<0.urn:uuid:D3D8A265C4834B582A11495537312963@apache.org>.
.
<?xml version='1.0' encoding='UTF-8'?>.
<soapenv:Envelope>.
....
....
<soapenv:Body>.
   <xenc:EncryptedData>.
      <xenc:CipherData>.
         <xenc:CipherValue><xop:Include href="cid:1.urn:uuid:D3D8A265C4834B582A11495537312911@apache.org"/>.
         </xenc:CipherValue>.
      </xenc:CipherData>.
   </xenc:EncryptedData>.
</soapenv:Body>.
</soapenv:Envelope>.
.
--MIMEBoundaryurn_uuid_D3D8A265C4834B582A11495537312962.
content-id:<1.urn:uuid:D3D8A265C4834B582A11495537312911@apache.org>.
content-type:application/octet-stream.
content-transfer-encoding:binary.
.
o Joooooo Io~(*obooeooo^o.
```

Sample 10 : Client Configuration

- sample10/client.axis2.xml
- Use of an Xpath expression to specify the base64 text nodes

```
<parameter name="OutflowSecurity">
    <action>.. Encrypt ..</action>
<optimizeParts>//xenc:EncryptedData/xenc
  :CipherData/xenc:CipherValue</optimizeP
  arts>
</parameter>
```

Is there any other way to set the configuration properties?

YES !!!

- A client can use two helper classes to generate the Axis2 inflow and outflow configuration parameters
- Parameters are made available via message context

Sample 11 : Client

- sample11/src/org/apache/rampart/samples/sample11/Client.java

- Set up the configuration parametes
  - getOutflowConfiguration()
  - getInflowConfiguration()

- client.engageModule(new QName("rampart"));

Outflow configuration

```
<parameter name="OutflowSecurity">
  <action>
    <items>Timestamp Signature Encrypt</items>
    <user>client</user>

 <passwordCallbackClass>org.apache.rampart.samples.sample11.PWCBHandler</passwordCallbackClass>
    <signaturePropFile>client.properties</signaturePropFile>
    <signatureKeyIdentifier>DirectReference</signatureKeyIdentifier>
    <encryptionKeyIdentifier>SKIKeyIdentifier</encryptionKeyIdentifier>
    <encryptionUser>service</encryptionUser>
  </action>
</parameter>
```

Outflow configuration - New

```
    OutflowConfiguration ofc = new OutflowConfiguration();

    ofc.setActionItems("Timestamp Signature Encrypt");

    ofc.setUser("client");


ofc.setPasswordCallbackClass("org.apache.rampart.samples.sample11.PWCBHandler"
);

    ofc.setSignaturePropFile("client.properties");

    ofc.setSignatureKeyIdentifier(WSSHandlerConstants.BST_DIRECT_REFERENCE);

    ofc.setEncryptionKeyIdentifier(WSSHandlerConstants.SKI_KEY_IDENTIFIER);

    ofc.setEncryptionUser("service");


    ofc.getProperty();
```

Inflow configuration

```
<parameter name="InflowSecurity">
  <action>
    <items>Timestamp Signature Encrypt</items>
<passwordCallbackClass>org.apache.rampart.samples.sample11.PWCBHandler</passwordCallbackClass>
    <signaturePropFile>client.properties</signaturePropFile>
  </action>
</parameter>
```

Inflow configuration - New

```
InflowConfiguration ifc = new InflowConfiguration();

ifc.setActionItems("Timestamp Signature Encrypt");

ifc.setPasswordCallbackClass("org.apache.rampart.samples.sample
11.PWCBHandler");

ifc.setSignaturePropFile("client.properties");


ifc.getProperty();
```

Future of Axis2 Security

- Rampart - WS-SecureConversation support
  - Can be used with Sandesha2 to provide secure-reliable communication
- Rahas - WS-Trust implementation for Axis2
- WSS4J 2.0 - WS-SecurityPolicy support

Stay in touch

- http://ws.apache.org/wss4j
- http://ws.apache.org/axis2

- wss4j-dev@ws.apache.org
- axis-dev@ws.apache.org
- axis-user@ws.apache.org

Thanks :-)

Any Questions?