JAX-WS Handler To Overcome JBWS-2640

When I tried Jboss Web Service (3.1.2 SP1 , which is delivered with JBoss AS 5.1), I meet the issue addressed in JBWS-2640 (https://jira.jboss.org/jira/browse/JBWS-2640). I review the source codes of JBossWS Native 3.2.1, the issue still exists.

Here is the generated SOAP Header

```
      <env:Header>
            <wsse:Security env:mustUnderstand='1'
                  xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
                  xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd'
                  xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd'>
                  <wsu:Timestamp wsu:Id='timestamp'>
                        <wsu:Created>2009-12-02T11:41:20.559Z
                        </wsu:Created>
                        <wsu:Expires>2009-12-02T11:46:20.559Z
                        </wsu:Expires>
                  </wsu:Timestamp>
                  <wsse:UsernameToken wsu:Id='token-3-1259754080559-1113559242'>
                        <wsse:Username>admin</wsse:Username>
                        <wsse:Password
                              Type='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordDigest'>P2PPAWUhA2eBf+4KmPgZNLEOU/8=
                        </wsse:Password>
                        <wsse:Nonce
                              EncodingType='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary'>YAfdMfdSHimIDfBocxBTWpLP1BtiOyl850HMrFWpxHw=
                        </wsse:Nonce>
                        <wsse:Created>2009-12-02T11:41:20.558Z
                        </wsse:Created>
                  </wsse:UsernameToken>
            </wsse:Security>
      </env:Header>
```

Here is the useCreated mentioned in WS-Security UsernameToken profile:

```
<wsse:UsernameToken wsu:Id="Example-1">
158 <wsse:Username> ... </wsse:Username>
159 <wsse:Password Type="..."> ... </wsse:Password>
160 <wsse:Nonce EncodingType="..."> ... </wsse:Nonce>
161 <wsu:Created> ... </wsu:Created>
162 </wsse:UsernameToken>
```

The namespace for useCreated is not correct.

The reason is the wrong namespace is set in org.jboss.ws.extensions.security.element.UsernameToken

```
public Element getElement()
```

```
   {
      if (cachedElement != null)
         return cachedElement;

      Element element = doc.createElementNS(Constants.WSSE_NS,
Constants.WSSE_PREFIX + ":" + "UsernameToken");
      element.setAttributeNS(Constants.WSU_NS, Constants.WSU_ID, getId());
      Element child = doc.createElementNS(Constants.WSSE_NS, Constants.WSSE_PREFIX
+ ":" + "Username");
      child.appendChild(doc.createTextNode(username));
      element.appendChild(child);
      child = doc.createElementNS(Constants.WSSE_NS, Constants.WSSE_PREFIX + ":" +
"Password");
      child.appendChild(doc.createTextNode(password));
      child.setAttribute("Type", digest ? Constants.PASSWORD_DIGEST_TYPE :
Constants.PASSWORD_TEXT_TYPE);
      element.appendChild(child);
      if (digest)
      {
         if (nonce != null)
         {
            child = doc.createElementNS(Constants.WSSE_NS, Constants.WSSE_PREFIX +
":" + "Nonce");
            child.appendChild(doc.createTextNode(nonce));
            child.setAttribute("EncodingType", Constants.BASE64_ENCODING_TYPE);
            element.appendChild(child);
         }
         if (created != null)
         {
            child = doc.createElementNS(Constants.WSSE_NS, Constants.WSSE_PREFIX +
":" + "Created");
            child.appendChild(doc.createTextNode(created));
            element.appendChild(child);
         }
      }
      cachedElement = element;
      return cachedElement;
   }
```

The impacts of the defect:

1. If the web service client and server both use JBossWS Native, the system works well, because the server and client runtime both use org.jboss.ws.extensions.security.element.UsernameToken to handle the UsernameToken Header. So the client or server runtime can retrieve the *useCreated* value.
2. If either of web service client or server doesn't use JBossWS Native, useCreated is in the namespace of WSU and it will not be retrieve the by WebService server and the authentication will always fail. So the platform independence of web service is broken. It is definitely limit the usage of JBossWS Native

I don't want to make the change and build a customized JBossWS Native.

So I use JAX-WS Handler to correct the SOAP header.

Here is the key part of the codes

```
protected boolean handleInboundMessage(SOAPMessageContext context) {
            try {
                  SOAPHeader head =  context.getMessage().getSOAPHeader();
                  XPath xPath = XPathFactory.newInstance().newXPath();
                  xPath.setNamespaceContext(new UserNameTokenNamespaceContext());

                  // xpath = "./wsse:Security/wsse:UsernameToken/wsu:Created";
                  String useCreatedExpr = String.format(
                              "./%1$s:%2$s/%1$s:%3$s/%4$s:%5$s", WSSE_PREFIX,
                              SECURITY_PART, USERNAMETOKEN_PART, WSU_PREFIX,
                              USECREATED_PART);

                  Node useCreated = (Node) xPath.evaluate(useCreatedExpr, head,
                              XPathConstants.NODE);
                  if (useCreated != null && useCreated instanceof SOAPElement) {
                        // standard wsu:Created found, which is going to be
converted to
                        // wsse:Create
                        logger
                                    .warn("Change wsu:Created -> wsse:Created, see
https://jira.jboss.org/jira/browse/JBWS-2640");
                        ((SOAPElement) useCreated).setElementQName(new QName(
                                    WSSE_NAMESPACE_URI, USECREATED_PART));

                  }
                  return true;
            } catch (SOAPException e) {
                  logger.error(e);
            } catch (XPathExpressionException e) {
                  logger.error(e);
            }
            return false;
      }

      @Override
      public boolean handleMessage(SOAPMessageContext context) {
            Boolean outbound = (Boolean) context
                        .get(MessageContext.MESSAGE_OUTBOUND_PROPERTY);
            if (outbound == null)
                  throw new IllegalStateException("Cannot obtain required
property: "
                              + MessageContext.MESSAGE_OUTBOUND_PROPERTY);

            return outbound ? handleOutboundMessage(context)
                        : handleInboundMessage(context);
      }

      protected boolean handleOutboundMessage(SOAPMessageContext context) {
            try {
                  SOAPHeader head = context.getMessage().getSOAPHeader();
                  XPath xPath = XPathFactory.newInstance().newXPath();
                  xPath.setNamespaceContext(new UserNameTokenNamespaceContext());
                  // xpath ="./wsse:Security/wsse:UsernameToken/wsse:Created";
                  String useCreatedExpr = String.format(
                              "./%1$s:%2$s/%1$s:%3$s/%1$s:%4$s", WSSE_PREFIX,
                              SECURITY_PART, USERNAMETOKEN_PART, USECREATED_PART);
                  Node useCreated = (Node) xPath.evaluate(useCreatedExpr, head,
                              XPathConstants.NODE);
                  if (useCreated != null && useCreated instanceof SOAPElement) {
                        // JBossWS generated wsse:Created found, which should be
```

```
                        // wsu:Create
                        logger
                                    .warn("Change wsse:Created -> wsu:Created, see
https://jira.jboss.org/jira/browse/JBWS-2640");
                        ((SOAPElement) useCreated).setElementQName(new QName(
                                    WSU_NAMESPACE_URI, USECREATED_PART));
                  }
                  return true;
            } catch (SOAPException e) {
                  logger.error(e);
            } catch (XPathExpressionException e) {
                  logger.error(e);
            }
            return false;
      }
```

On the client side, I add jaxws-client-config.xml and jboss-wsse-client.xml to the META-INF of the client jar

I create a customized client config:

```
      <client-config>
            <config-name>[JBWS-2640] Free WSSecurity Client</config-name>
            <post-handler-chains>
                  <javaee:handler-chain>
                        <javaee:protocol-bindings>##SOAP11_HTTP
                              ##SOAP11_HTTP_MTOM</javaee:protocol-bindings>
                        <javaee:handler>
                              <javaee:handler-name>WSSecurityHandlerOutbound
                              </javaee:handler-name>
                              <javaee:handler-class>

      org.jboss.ws.extensions.security.jaxws.WSSecurityHandlerClient
                              </javaee:handler-class>
                        </javaee:handler>
                        <javaee:handler>
                              <javaee:handler-name>WSSEUsernameTokenHandler
                              </javaee:handler-name>
                              <javaee:handler-class>

      hxie.framework.webservice.jbossws.handler.WSSEUsernameTokenHandler
                              </javaee:handler-class>
                        </javaee:handler>
                  </javaee:handler-chain>
            </post-handler-chains>
      </client-config>
```

WSSecurityHandlerOutbound will generate the SOAP security header. So WSSEUsernameTokenHandler must be executed after WSSecurityHandlerOutbound. WSSEUsernameTokenHandler must follow WSSecurityHandlerOutbound in the configuration file because the handler execution order is from top to down in the client side.

When a port instance is created, I add the following codes to ask the runtime to use the customized client config:

```
String  configName="[JBWS-2640] Free WSSecurity Client";
String  configFile="META-INF/jaxws-client-config.xml";
StubExt stub = (StubExt) port;
stub.setConfigName(configName, configFile);
```

Here is a sample of corrected SOAP Header

```
      <soapenv:Header>
            <wsse:Security
                  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
                  <wsse:UsernameToken wsu:Id="UsernameToken-2"
                        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
                        <wsse:Username>admin</wsse:Username>
                        <wsse:Password
                              Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordDigest">x1DXXGGQtHdvxot6xE70Js2ahlg=
                        </wsse:Password>
                        <wsse:Nonce
                              EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">akhPzGxGwuAKyiby9H4jIw==
                        </wsse:Nonce>
                        <wsu:Created>2009-12-02T14:06:49.705Z
                        </wsu:Created>
                  </wsse:UsernameToken>
            </wsse:Security>
      </soapenv:Header>
```

On the server side,I add jboss.xml, jaxws-endpoint-config.xml and jboss-wsse-server.xml to the META-INF of Web Service EJB jar.

I created a customized endpoint config:

```
    <endpoint-config>
    <config-name>[JBWS-2640] Free WSSecurity Endpoint</config-name>
    <post-handler-chains>
      <javaee:handler-chain>
        <javaee:protocol-bindings>##SOAP11_HTTP
##SOAP11_HTTP_MTOM</javaee:protocol-bindings>
       <javaee:handler>
          <javaee:handler-name>WSSecurity Handler</javaee:handler-name>
          <javaee:handler-
class>org.jboss.ws.extensions.security.jaxws.WSSecurityHandlerServer</javaee:handle
r-class>
        </javaee:handler>
        <javaee:handler>
          <javaee:handler-name>WSSEUsernameTokenHandler</javaee:handler-name>
          <javaee:handler-
class>hxie.framework.webservice.jbossws.handler.WSSEUsernameTokenHandler</javaee:ha
ndler-class>
        </javaee:handler>
        <javaee:handler>
          <javaee:handler-name>Recording Handler</javaee:handler-name>
          <javaee:handler-
class>org.jboss.wsf.framework.invocation.RecordingServerHandler</javaee:handler-
```

```
class>
        </javaee:handler>

      </javaee:handler-chain>
    </post-handler-chains>
  </endpoint-config>
```

In *WSSecurity Handler,* username, password, nonce, usecreated will be set to UsernameCallBack. So *WSSEUsernameTokenHandler* must be executed before *WSSecurity Handler*.
WSSEUsernameTokenHandler must follow *WSSecurity Handler* in the configuration file because the handler execution order is from down to client in the client side, which is specified in 9.2.1.2 Handler Ordering in JAX-WS 2.1 specification.

Specify the config name and config file in jboss.xml so that the JBossWS Native could use the handler
The web service part of jboss.xml looks like

```
<webservices>
            <context-root>/hxie/framework/security/admin/logic
            </context-root>
            <webservice-description>
                  <webservice-description-name> security logic web service
                  </webservice-description-name>
                  <config-name>[JBWS-2640] Free WSSecurity Endpoint
                  </config-name>
                  <config-file>META-INF/jaxws-endpoint-config.xml
                  </config-file>
            </webservice-description>
</webservices>
```

P.S.
When JBossWS Team fix JBWS-2640,
the *WSSEUsernameTokenHandler* must be removed from the configuration.
On the client side, it is recommanded to remove the configuration. If not, it will not cause correctness issue.