Invoking a service secured with Username token, using Jmeter

Here the test scenario is to invoke a pass through proxy service in ESB which is secured with Username Token(But this can be used to invoke and load test any kind of web service which is secured with Username token).

The difficulty in invoking services secured with UT using Jmeter is that, Jmeter does not embed the security headers required in the SOAP request. What Jmeter does is send the SOAP request we give with concurrent threads.

Following is a typical SOAP request with username token headers. (I copied the soap message from Soap tracer tool)
Note the **Timestamp values ('created' and 'expired')**, and the **NONCE** value which is randomly generated using the current time, and another **timestamp (**wsu:Created**)** at the end. Issue with Jmeter is, it does not generate those values. If the NONCE and Timestamp values are duplicated in subsequent requests ESB will reject to invoke the service thinking this is a malicious attack.

```
<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">
         <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" wsu:Id="Timestamp-215">
            <wsu:Created>2012-01-13T08:29:59Z</wsu:Created>
            <wsu:Expires>2012-01-13T08:30:01Z</wsu:Expires>
         </wsu:Timestamp>
            <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
            wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-214">
            <wsse:Username>admin</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
                        profile-1.0#PasswordText">admin</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
            security-1.0#Base64Binary">Te03IAKHD3JVvd3VayYosyHBkkFcSGJgLc5vGMKRGtc=</wsse:Nonce>
           <wsu:Created>2012-01-13T13:59:59.254Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body />       – change the soap Body here depending on your service input parameters, in this case no parameters are passed
</soapenv:Envelope>
```

Following is the SOAP request that I used in Jmeter, with those security element parameterized. Copy this into your Jmeter test plan as the SOAP-XML request.

SOAP message:

```
<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">
         <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-215">
            <wsu:Created>#timeC#</wsu:Created>
            <wsu:Expires>#timeE#</wsu:Expires>
         </wsu:Timestamp>
         <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-214">
            <wsse:Username>admin</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">#nonce#</wsse:Nonce>
            <wsu:Created>#UTtimeC#</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body />
</soapenv:Envelope>
```

So the solution is to generate those values using a Java code. And then embed those values into the (parameters of) soap request headers Using the **Beanshell Preprocessor** element in Jmeter *(In Jmeter right click on the Thread group → Add → PreProcessors → Beanshell Preprocessor)*.

Following Java code snippet can be used to generate those above mentioned Timestamps and NONCE values. Copy this code inside the 'Script' text box in BeanShell Preprocessor, in Jmeter. Here some external libraries are used for generating NONCE, therefore copy the '**jboss-common.jar**' and '**jbossws-core.jar**' into **JMETER_HOME/lib** folder (and restart before running the test).

So this Java code will generate the NONCE and TimeStamp values in each concurrent thread, and will embed those values in each respective SOAP request. So you can do a load test with any number of concurrent requests.

Beanshell prepocessor Java code:

```
import org.apache.jmeter.protocol.http.sampler.SoapSampler;
import org.jboss.ws.extensions.security.nonce.DefaultNonceGenerator;     // (jbossws-core.jar)
import org.jboss.util.Base64;    // this class is called inside the DefaultNonceGenerator class (jboss-common.jar)
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

            // generating NONCE
        DefaultNonceGenerator nonce_Gen = new DefaultNonceGenerator();
        String nonce = nonce_Gen.generateNonce();

           // generating relevant Timestamps
        long ctmilli = System.currentTimeMillis();   // current time in milliseconds
        SimpleDateFormat dformat1 = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");    // timestamp
format with nonce
        SimpleDateFormat dformat2 = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");   //timestamp
format for usernametoken, with miliseconds
        Date dtime = new Date();
        dformat1.setTimeZone(TimeZone.getTimeZone("UTC"));

        dtime.setTime(ctmilli);   // current time
        String timeCreated =  dformat1.format(dtime);      // timestamp created in format1
        String uttimeCreated =  dformat2.format(dtime);   // timestamp created in format2
        dtime.setTime(ctmilli+2000);   // setting the timeout  for 2seconds,  change the timeout as required
        String timeExpire =  dformat1.format(dtime);       // expiration timestamp in format1

// to embed the security elements into
SoapSampler soapSampler = (SoapSampler) sampler;        // get the sampler object
String modRequest= soapSampler.getXmlData().replaceFirst("#timeC#", timeCreated);  // read the given soap
modRequest = modRequest.replaceFirst("#timeE#", timeExpire);          // modify the given soap request
modRequest = modRequest.replaceFirst("#nonce#", nonce);                          // - with relevant headers
modRequest = modRequest.replaceFirst("#UTtimeC#", uttimeCreated);
soapSampler.setXmlData(modRequest);             // set the modified soap msg into the sampler
```

- Nirodha