Transforming a SOAP Message

Duration: 5 Mins

This tutorial will illustrate how to use Smooks to move and reformat data within an XML document.

In this tutorial, we will transform a "findAddresses" SOAP action message from Acme. All Acme SOAP requests have the security credentials buried within the SOAP body as a standard contract paramater. The SOAP endpoint to service this contract requires that the security credentials comply with the WS-Security standard. This means that the Acme SOAP message needs to be transformed to:

1. Remove the Credentials from the SOAP body.
2. Add the Credentials to the SOAP header in compliance with the WS-Security standard.

To perform this transformation we will use:

1. The Smooks JavaBean Cartridge to extract credentials information from the SOAP body.
2. The Smooks Misc Cartridge to remove the credentials DOM elements from the SOAP body.
3. The Smooks Templating Cartridge to generate the WS-Security block an add it to the SOAP header.

Transformation Requirements

The Input SOAP Request:
Note that the security credentials are buried in the SOAP body.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:m0="http://schemas.acme.com/addressfinder/"
   xmlns:m1="http://schemas.acme.com/addressmanager/"
   xmlns:wsa="http://www.w3.org/2005/03/addressing">

  <SOAP-ENV:Header>
    <wsa:MessageID>9ccdc110-b5e1-11da-bb6e-e2761a0ce10a</wsa:MessageID>
    <wsa:Action>urn:x-acme:servicetype:Address:findAddresses</wsa:Action>
  </SOAP-ENV:Header>

  <SOAP-ENV:Body>
    <m:findAddresses xmlns:m="v2"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <n1:acmeCreds xmlns:n1="http://schemas.acme.com/security/"
xsi:type="n1:AcmeCreds">
        <n1:usr>johnnym</n1:usr>
        <n1:pwd>guessagain</n1:pwd>
      </n1:acmeCreds>
      <addresses xsi:type="SOAP-ENC:Array"
SOAP-ENC:arrayType="m0:Address[1]">
        <m0:item0 xsi:type="m0:Address">
          <houseNumber>380</houseNumber>
          <street>New York St.</street>
          <city>Redlands</city>
          <state_prov>CA</state_prov>
          <zone>92373</zone>
          <country>US</country>
        </m0:item0>
        <m0:item1 xsi:type="m0:Address">
          <street>1000 Main St.</street>
          <country>US</country>
        </m0:item1>
      </addresses>
      <addressManagerOptions xsi:type="m1:AddressManagerOptions">
        <dataSource xsi:type="xsd:string">GDT.Address.US</dataSource>
      </addressManagerOptions>
      <token xsi:type="xsd:string">MyToken</token>
    </m:findAddresses>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

The "Required" Transformed SOAP Request:
The following shows the WS-Security related transformations that are required for the endpoint. Note that other transformations will probably need to be made to this message, but these are not being considered in this tutorial.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:m0="http://schemas.acme.com/addressfinder/"
   xmlns:m1="http://schemas.acme.com/addressmanager/"
   xmlns:wsa="http://www.w3.org/2005/03/addressing">

  <SOAP-ENV:Header>
    <wsa:MessageID>9ccdc110-b5e1-11da-bb6e-e2761a0ce10a</wsa:MessageID>
    <wsa:Action>urn:x-acme:servicetype:Address:findAddresses</wsa:Action>
    <wsse:Security
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
      <wsse:UsernameToken>
        <wsse:Username>johnnym</wsse:Username>
        <wsse:Password>guessagain</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </SOAP-ENV:Header>

  <SOAP-ENV:Body>
    <m:findAddresses xmlns:m="v2"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

      <addresses xsi:type="SOAP-ENC:Array"
SOAP-ENC:arrayType="m0:Address[1]">
        <m0:item0 xsi:type="m0:Address">
          <houseNumber>380</houseNumber>
          <street>New York St.</street>
          <city>Redlands</city>
          <state_prov>CA</state_prov>
          <zone>92373</zone>
          <country>US</country>
        </m0:item0>
        <m0:item1 xsi:type="m0:Address">
          <street>1000 Main St.</street>
          <country>US</country>
        </m0:item1>
      </addresses>
      <addressManagerOptions xsi:type="m1:AddressManagerOptions">
        <dataSource xsi:type="xsd:string">GDT.Address.US</dataSource>
      </addressManagerOptions>
      <token xsi:type="xsd:string">MyToken</token>
    </m:findAddresses>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Create the "Credentials" JavaBean

We create the Credentials JavaBean to store the Acme credentials (username and password) captured by the Smooks JavaBean Cartridge from the SOAP body during the Assembly Phase. The Smooks JavaBean Cartridge will then make this bean information available to the Smooks Templating Cartridge to generate the WS-Security block to be added to the SOAP header.

The Bean:

```
public class Credentials {

    private String username;
    private String password;

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Create the StringTemplate Template

We need to write a template for generating the WS-Security block to be added to the SOAP header.

The Smooks Templating Cartridge makes the Credentials JavaBean information (gathered by Smooks JavaBean Cartridge) available to the String Template templating engine - see StringTemplateContentDeliveryUnitCreator. Once the StringTemplateContentDeliveryUnitCreator applies the template (via StringTemplate), it takes the generated content (the WS-Security block in this case) and adds it to the document.

WsseCredsWriter.st:

```
<wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
    <wsse:UsernameToken>
        <wsse:Username>$credentials.username$</wsse:Username>
        <wsse:Password>$credentials.password$</wsse:Password>
    </wsse:UsernameToken>
</wsse:Security>
```

Write the Smooks Configurations

We need to create Smooks configurations for populating the Credentials JavaBean and applying the StringTemplate Template i.e. for performing the transformation.

Note, refer to the SmooksResourceConfiguration and AbstractBeanPopulator docs when reading these configurations.

acme-creds.cdrl:

```
<?xml version="1.0"?>
<!DOCTYPE smooks-resource-list PUBLIC "-//MILYN//DTD SMOOKS 1.0//EN"
```

```
"http://www.milyn.org/dtd/smooksres-list-1.0.dtd">

<smooks-resource-list default-useragent="acme-request">

 <!--
=========================================================================
  Configure the Credentials bean creation and population.
  In the case of a SOAP invocation from "Acme", Acme insist
  on setting the username/password details in the SOAP body.
  We need to extract the user creds from the body during the assembly
  phase and have it available for a template that gets applied during the
  processing phase - adding the security info to the header as a
  WS-Security block.
 =========================================================================
-->
 <smooks-resource selector="acmeCreds"
    namespace="http://schemas.acme.com/security/"
    path="org.milyn.javabean.AssemblyPhaseBeanPopulator">
  <!-- create and bind the Credentials bean instance on encountering the
acmecreds element. -->
  <param name="beanId">credentials</param>
  <param
name="beanClass">org.milyn.templating.stringtemplate.acmesecsample.Credent
ials</param>
 </smooks-resource>
 <smooks-resource selector="acmeCreds usr"
    namespace="http://schemas.acme.com/security/"
    path="org.milyn.javabean.AssemblyPhaseBeanPopulator">
  <!-- extract the username and set it on the Credentials bean -->
  <param name="beanId">credentials</param>
  <param name="setterName">setUsername</param>
 </smooks-resource>
 <smooks-resource selector="acmeCreds pwd"
    namespace="http://schemas.acme.com/security/"
    path="org.milyn.javabean.AssemblyPhaseBeanPopulator">
  <!-- extract the password and set it on the Credentials bean -->
  <param name="beanId">credentials</param>
  <param name="setterName">setPassword</param>
 </smooks-resource>

 <!-- Remove the acmecreds element from the SOAP body.  In reality, you'd
   prob not need to remove this element in this way. It would probably
   get removed as part of a more general revamp of the SOAP body i.e. to
   "make it fit" a different schema altogether. (See the Misc Cartridge)-->
 <smooks-resource selector="acmeCreds"
    namespace="http://schemas.acme.com/security/"
    path="org.milyn.cdres.assemble.RemoveElementAU" />

 <!--
=========================================================================
  Use a StringTemplate template to write the Credentials info (gathered
  during the Assembly Phase) into the SOAP header (add them to
  the header i.e. "addto").
```

```
  Note: This template is applied during the "Processing Phase".
 ========================================================================
-->
 <smooks-resource selector="Envelope Header"
     namespace="http://schemas.xmlsoap.org/soap/envelope/"

path="/org/milyn/templating/stringtemplate/acmesecsample/WsseCredsWriter.s
t">
  <!-- add the template to the header -->
  <param name="action">addto</param>
 </smooks-resource>
```

```
</smooks-resource-list>
```

Write the code to Executing the Transformation

```
SmooksStandalone smooks = new SmooksStandalone("UTF-8");

// Configure Smooks...
smooks.registerUseragent("acme-findAddresses-request", new String[]
{"acme-request"});
TemplatingUtils.registerCDUCreators(smooks.getContext());
smooks.registerResources("acme-creds",
getClass().getResourceAsStream("acme-creds.cdrl"));

// Perform the transformation...
InputStream requestStream =
getClass().getResourceAsStream("AcmeFindaddressRequest.xml");
String requestResult =
smooks.filterAndSerialize("acme-findAddresses-request", requestStream);
System.out.println(requestResult);
```