17 March 2013

NIEM Web Services API

Version 1.0

URI: http://reference.niem.gov/niem/specification/web-services-api/1.0/

Change History

No.	Date	Reference: All, Page, Table, Figure, Paragraph	A = Add. M = Mod. D = Del.	Revised By	Change Description
0.1	11/25/2009	All	A	Chris Lewis	Initial version
0.2	9/14/2011	Page 5	M	John Matthews	Added parameter
0.3	3/17/2013	Page 6	C	John Matthews	Removed conformance

Contents

1 Introduction

NIEM now offers access to some of the functionality that currently exists on the NIEM Web site through the use of SOAP Web Services. This will allow the ecosystem of NIEM tools and users to grow as more complex and useful features can be integrated into tools and make the lives of users easier. Consider these two use cases:

- A user may now write a program to generate subsets of NIEM by providing the wantlist to the web service and get back the generated subset, rather than manually generate a subset in the NIEM SSGT web application.
- A user may now write a program to access the Subset Generation Web Service to search for NIEM components through a custom user interface, rather than use the interface provided by the Subset Generation Tool.

NIEM now offers three different classes of queries through Web Services (more will likely be added in the future). Each class of query is described in more detail within this document. NIEM requires that all requests sent to the Web Service API be signed. The "Web Service Security" section details the requirements and process for signing your API requests.

2 Web Service Security

NIEM utilizes WS-Security (Web Services Security: SOAP Message Security) for authenticating requests sent to the NIEM Web Services API and requires that all requests sent to the Web Service API be signed. NIEM supports version 1.0 of the WS-Security specification (available on the OASIS Web site – http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss).

WS-Security provides a single SOAP header, <Security>, for use in transmitting authentication information within the SOAP request. It provides the implementer choices for the specific signature method to use. NIEM utilizes X.509 certificates for this purpose. The X.509 certificates provide a mechanism for transmitting a public key (This public key, and the associated private key are components of Public-key cryptography. For information and terminology related to the cryptography, see http://en.wikipedia.org/wiki/Public-key_cryptography). To create a signed SOAP message, WS-Security specifies that you sign the request with the private key associated with the X.509 certificate, and then attach the X.509 certificate to the request by representing it as a BinarySecurityToken as defined in the WS-Security X.509 token profile specification (available on the OASIS Web site – http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss).

In order to sign the requests, you must have a valid X.509 certificate. Appendix A details the steps to generate and receive a valid certificate that has been signed by the trusted NIEM Certificate Authority. You must follow these steps before any requests to the Web Service API can be generated. If you just wish to verify that the X.509 certificate you received is valid and working, the tool SOAP-UI can be used to verify this. Please see Appendix B for a short overview of verifying the validity of your X.509 certificate using SOAP-UI. If a request is sent to the Web Service API without a signature, the SOAP response will state:

“com.sun.xml.wss.XWSSecurityException: Message does not conform to configured policy [SignaturePolicy(P)]: No Security Header found”

Once you have received your certificate, you may begin to generate requests. Each request sent to NIEM must be signed with the private key associated with your X.509 certificate. To create the signature, sign the Body element. At this point, requests sent to NIEM should be receiving responses. Several toolkits exist for many languages to automate the procedure of generating and attaching signatures that follow the WS-Security specification.

The following example is a header for a valid signed request. This example illustrates how NIEM makes use of WS-Security, and which configuration settings NIEM uses.

```
<soapenv:Envelope xmlns:sch="http://niem.gov/ws/schemas"
  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">
          <wsse:BinarySecurityToken EncodingType="http://docs.oasis-
            open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
            1.0#Base64Binary" ValueType="http://docs.oasis-
            open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
            1.0#X509v3" wsu:Id="CertId-10160805"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
            200401-wss-wssecurity-utility-1.0.xsd">
            <!--Your token here -->
          </wsse:BinarySecurityToken>
          <ds:Signature Id="Signature-27761480"
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
             <ds:SignedInfo>
                <ds:CanonicalizationMethod
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                <ds:SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
                  sha1"/>
                <ds:Reference URI="#id-21823376">
                   <ds:Transforms>
                      <ds:Transform
                        Algorithm="http://www.w3.org/2001/10/xml-exc-
                        c14n#"/>
                   </ds:Transforms>
                   <ds:DigestMethod
                     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/
                     >
                   <ds:DigestValue>DXwRPtVIYHjScTRU/2gV0dJa8tI=</ds:Dige
                     stValue>
                 </ds:Reference>
             </ds:SignedInfo>
             <ds:SignatureValue>DBBTVWc8f/iDFzkO1q/JIAESz/AkUB5rldbrmwdg
              yVVaksf4u/KXOONsHKdwsqzNBS0fC/C/Lp61bNTL51lgkCT3E4xgp+XT2
              QWbYVUrwBR0yXxaIAfZtiixu4Hi8fDGrYuZF8ZvvS7BfjecREB1yW5I5s
              GAGjPG72EKezXVJwY=
             </ds:SignatureValue>
             <ds:KeyInfo Id="KeyId-12821819">
                <wsse:SecurityTokenReference wsu:Id="STRId-9749991"
                  xmlns:wsu="http://docs.oasis-
                  open.org/wss/2004/01/oasis-200401-wss-wssecurity-
                  utility-1.0.xsd"><wsse:Reference URI="#CertId-
                  10160805" ValueType="http://docs.oasis-
```

```
                    open.org/wss/2004/01/oasis-200401-wss-x509-token-
                    profile-1.0#X509v3"/>
                  </wsse:SecurityTokenReference>
               </ds:KeyInfo>
            </ds:Signature>
            </wsse:Security>
   </soapenv:Header>
   <soapenv:Body wsu:Id="id-21823376" xmlns:wsu="http://docs.oasis-
      open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
```

A few things worth noting:

- EncodingType for the BinarySecurityToken is set to use WS-Security, specifically http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary
- ValueType for the BinarySecurityToken element is set to be X.509 certificates, specifically http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3
- The BinarySecurityToken should contain the base64 encoding of your X.509 certificate (place this where "<!-- Your token here -->" is in the example)
- The KeyInfo element must contain as SecurityTokenReference element, which must contain a Reference element. This Reference element must have a URI attribute which identifies the local BinarySecurityToken containing the X509 certificate
- The signed element (Body, in the case of NIEM) must contain a wsu:Id attribute

The above example has demonstrated the use of a Web Service header. In order to be more concise, example queries shown in the sections that follow will not contain Web Service headers. However, Web Service headers are always required in practice.

3 NIEM Web Services

The NIEM tool Web site currently exposes four Web Services:

- Search – accepts a query and returns the NIEM component or components matching the query.
- Subset Generation – accepts a wantlist and returns the subset schema for the given wantlist.
- Code List Generation – accepts a Microsoft Excel spreadsheet and returns a code list schema.

3.1 Search Web Service

The Web Services Description Language (WSDL) for the Search Web Service is located at:

http://niem.gtri.gatech.edu/niemtools/ws/search/search.wsdl

This service has two separate queries: one to search for components and one to retrieve a specific component.

3.1.1 Search for Components

The search service has two input parameters:

1. Type - The type of component to search for: "Property", "Type", or "Namespace".
2. SearchString - The query you wish to search for, i.e. the name of the component.

The service returns one element:

SearchResponse - A set of elements matching the type and the string from the query specified in the input.

The following is a sample call to the service:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nm="http://niem.gov/ws/schemas">
  <soapenv:Header/>
   <soapenv:Body>
      <nm:SearchRequest>
         <nm:Type>
            <nm:Value>Property</nm:Value>
         </nm:Type>
         <nm:SearchString>
            <nm:Value>Person</nm:Value>
         </nm:SearchString>
      </nm:SearchRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

3.1.2 Retrieve a Component

Once a user knows what component he/she wants, a second query is available to retrieve that component.

The retrieval service has one input parameter:

ID - The qualified Name of the component to retrieve; e.g., "nc:Person".

The service returns one element.

GetElementResponse - The XML representation of the component the user chooses to retrieve.

The following is a sample call to the service:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nm="http://niem.gov/ws/schemas">
   <soapenv:Header/>
   <soapenv:Body>
```

```
        <nm:GetElementRequest>
           <nm:ID>
              <nm:Value>nc:Person</nm:Value>
           </nm:ID>
        </nm:GetElementRequest>
     </soapenv:Body>
  </soapenv:Envelope>
```

3.2 Subset Generation Web Service

The WSDL for the Subset Generation Web Service is located at:

http://niem.gtri.gatech.edu/niemtools/ws/schemagenerator/genschema.wsdl

The service has two input parameters:

1. WantList - A base64 encoding of an XML Wantlist file
2. IncludeWantlist - A boolean to indicate whether to include the wantlist in the generated schema
3. IncludeDocumentation - A boolean to indicate whether to include the documentation in the generated schema

The service returns one element:

Response - A base64 encoding of the generated Subset

The following is a sample call to the service:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nm="http://niem.gov/ws/schemas">
   <soapenv:Header/>
   <soapenv:Body>
      <nm:GenerateSchemaRequest>
         <nm:WantList>
              <nm:DataFile>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTg
                 iPz4KPHc6V2FudExpc3QgdzpyZWxlYXNlPSIyLjAiIHc6cHJvZHVjdD0i
                 TklFTSIKICB4bWxuczphbnNpLW5pc3Q9Imh0dHA6Ly9uaWVtLmdvdi9ua
                 WVtL2Fuc2ktbmlzdC8yLjAiIHhtbG5zOnc9Imh0dHA6Ly9uaWVtLmdvdi
                 9uaWVtL3dhbnRsaXN0LzIiPgogIDx3OkVsZW1lbnQgdzpuYW1lPSJhbnN
                 pLW5pc3Q6RmFjZUltYWdlRXllQ29sb3JBdHRyaWJ1dGUiIHc6aXNSZWZl
                 cmVuY2U9ImZhbHNlIi8+CjwvdzpXYW50TGlzdD4K</nm:DataFile>
         </nm:WantList>
         <nm:IncludeWantList>
            <nm:Value>True</nm:Value>
         </nm:IncludeWantList>
         <nm:IncludeDocumentation>
            <nm:Value>True</nm:Value>
         </nm:IncludeDocumentation>
      </nm:GenerateSchemaRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

3.3 Code List Generation Web Service

The WSDL for Code List Generation Web Service can be found at:

http://niem.gtri.gatech.edu/niemtools/ws/codelistgenerator/gencodelist.wsdl

The service has six input parameters:

1. CodeList - A base64 encoding of the Excel version of the code list
2. Prefix - The prefix of the namespace for the code list
3. URI - The URI of the namespace for the code list
4. Location - The relative path of the schema file to use in the returned zip
5. Version - The version of the namespace to use in the schema
6. Definition – The definition of the namespace to use in the schema

The service returns one element:

Response - A base64 encoding of the generated XML version of the code list

The following is a sample call to the service:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nm="http://niem.gov/ws/schemas">
   <soapenv:Header/>
   <soapenv:Body>
      <nm:GenerateCodeListRequest>
         <nm:CodeList>
            <nm:DataFile>
            ** Replace this with Base64 Encoding of Code List here **
            </nm:DataFile>
         </nm:CodeList>
         <nm:Prefix>
            <nm:Value>test</nm:Value>
         </nm:Prefix>
         <nm:URI>
            <nm:Value>http://niem.gov/test/test</nm:Value>
         </nm:URI>
         <nm:Location>
            <nm:Value>local/codes/schema.xsd</nm:Value>
         </nm:Location>
         <nm:Version>
            <nm:Value>2</nm:Value>
         </nm:Version>
         <nm:Definition>
            <nm:Value>Our definition</nm:Value>
         </nm:Definition>
      </nm:GenerateCodeListRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Certificate Creation

The following is the recommended and supported way to generate a X.509 certificate and have it signed by the NIEM Certificate Authority. This requires that OpenSSL (http://www.openssl.org/) be installed on your system. The examples below use Unix syntax; however, a user can also generate the key using similar commands in Windows or other operating systems. These instructions generate a certificate for use in a Java tool; however, the certificate created can be used in any language or tool.

Begin by creating a working directory and change to this directory

```
$ mkdir ExampleCertificateRequest

$ cd ExampleCertificateRequest/
```

Next generate the private key and associated public key. In the command below, replace "requestingClient" with the alias of your choice.

```
$ keytool -genkey -alias requestingClient -keyalg RSA -keystore
exampleClientKeystore.jks
```

Create a password. Fill out the requested details. For the question "What is your first and last name" (or commonName, if presented with that text), enter the name of the tool which will be using the generated certificate.

At this point, the certificate is unsigned. A certificate signing request must be generated so that the NIEM Certificate Authority can sign it.

```
$ keytool -certreq -keystore exampleClientKeystore.jks -alias
requestingClient -file requestingClient.cert.req
```

The file generated (requestingClient.cert.req) is the certificate signing request. Send this file to pgmw-system@gtri.gatech.edu with the subject "NIEM Web Service Certificate Request".

Within a couple of business days, you will receive an email back with two files attached. One is your signed certificate (this file will be called signedRequestingClient.cert); the other is the public key for the NIEM Certificate Authority (this file will be called cacert.cert).

Place these two files in your working folder. Import the NIEM public key into the keystore.

```
$ keytool -import -file cacert.cert -alias NIEMWebService -keystore
exampleClientKeystore.jks
```

Hit "Y" when prompted to trust the certificate. Now your signed certificate can be loaded.

```
$ keytool -import -file signedRequestingClient.cert -alias
requestingClient -keystore exampleClientKeystore.jks
```

The signed certificate for your tool has now been imported into the keystore, and is ready for use.

Appendix A: Testing with SOAPUI

SOAP-UI (http://www.soapui.org/) is a free, open source Web Services testing tool created by Eviware. It can be used to quickly generate SOAP requests to verify Web Service connections, settings, and configurations. A user can employ SOAP-UI to verify that he/she is able to connect to the NIEM Web Services and generate a valid signature with the X.509 certificate that he/she requested and received from the NIEM Certificate Authority.

A full SOAP-UI tutorial is outside the scope of this document. SOAP-UI provides a detailed walk-through (with screenshots) of testing WS-Security requests here: http://www.soapui.org/userguide/projects/wss.html. The reader should review or already be familiar with the content of this tutorial before continuing with the rest of this section.

The following briefly outlines the steps for configuring the security and signature settings required SOAP-UI to communicate with the NIEM Web Services:

1. Create your keystore within SOAP-UI.
2. Add an Outgoing WSS Configuration of type Signature.
3. Use the “requestingClient” alias, rather than “niemwebserver” alias.
4. Change “Key Identifier Type” to “BinarySecurityToken”.
5. Leave “Signature Algorithm” and “Signature Canonicalization” set to default.
6. Check the box next to “Use Single Certificate”.
7. At this point, save your project within SOAP-UI.
8. If a request is already open, close it.
9. Open a request (or generate a new one using one of the WSDL links from the NIEM Web Services).
10. Select the “Auth” tab at the bottom of the Request window.
11. Select the option to choose the security configuration just created under “Outgoing WSS.”
12. The request should now be configured to generate and attach a signature. Verify this by viewing the raw data of the request sent to ensure that a <wsse:Security> element was added within the <soapEnv:Header> element.

NIEM.gov