▼ ▼ ▼ ▼ ▼ ▼ ▼ ▼ ▼ ▼ ▼

CANNEX

Web-Services Security
Version 1.04

▲ ▲ ▲ ▲

November 2018

CANNEX Financial Exchanges Limited
1200 Bay Street, Suite 1001
Toronto, Ontario
Canada, M5R 2A5

Telephone: (416) 926-0882
Toll Free: 1-800-387-1269
Fax: (416) 926-0706

www.cannex.com
cannex@cannex.com

Description

Cannex's Web Service Security requires that an SSL transport be used, that the Web Service is invoked by a POST method, and either one of the WS-Security profiles are used; Username Token Profile 1.1 or the X.509 Certificate Token Profile 1.1. Since each WS-Security profile provides for a number of different configurations (features) that will adhere with the specification, the following sections will go into specific details to clarify which of these configurations are required.

Username Token Profile 1.1

Reference: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf

A Cannex compliant Username Token Profile 1.1 must be constructed as the following SOAP Header example shows:

```
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-1">
      <wsse:Username>wernerd</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">mDyN3ZYwGBSYA7nNrSVQbVqySH8=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">oWKh3qJUOqKS4JP5e1IcPg==</wsse:Nonce>
      <wsu:Created>2012-07-19T19:33:03.009Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>
```

Element values that will be provided at runtime are Username, Password, Nonce and Created. The wsu:Id value "UsernameToken-1" of the element UsernameToken can be supplied with a different value as long as it complies with XML Id value rules.

As specified in the reference document, the Password value is calculated as:

```
Password_Digest = Base64 ( SHA-1 ( nonce + created + password ) )
```

Thus the Password_Digest of the above example used the Base64 encoded nonce value "`oWKh3qJUOqKS4JP5e1IcPg==`", the created value "`2012-07-19T19:33:03.009Z`" and the plain text password value "`verySecret`" to generate the value of "`mDyN3ZYwGBSYA7nNrSVQbVqySH8=`". Since encryption is done at the byte level; the following pseudo code represents such an operation:

```
BASE64_ENCODE( SHA1( BASE64_DECODE(nonce) + BYTES_UTF8(created) + BYTES_UTF8(password) ) )
```

The Nonce value is a "use once" globally unique Base64 encoded value.

The Created value must be in UTC format (as in the example above) so that Time zone differences can be resolved correctly. The Created value is considered valid if it resolves within plus or minus 150 seconds of Cannex's server time (i.e. a 300 second window). Cannex's server time can be viewed at the following URL: https://www.cannex.com/app/CANX/Syscheck

And finally, please note that both the unique Username and the plain text version of the password value will be issued by Cannex.

Appendix A contains six more examples for further verification.

Appendix B contains coding information to help construct the security header.

X.509 Certificate Token Profile 1.1

Reference: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf

A Cannex compliant X.509 Certificate Token Profile 1.1 must be constructed as the following SOAP Header example shows:

```
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-2">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="S"/>
        </ds:CanonicalizationMethod>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference URI="#TS-1">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="wsse
S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>uH3nd+17uMbQWjvUSXt50RPmGtw=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#BODY-1">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList=""/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>q1kpy3CGZ5L4N60LrXaYhvw6v5Q=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
<ds:SignatureValue>mPnKNrsR86dJu7Cd4Cz1n7iUswD6GUqHgd82AS7xEbiXN3Netu/ameomvLCD4ncy39gvGRGK3kxu
gb+f0yUcqQk1b72j5+JqyJRyYcrMEg+cgHF3BNkzLFtFyPE83mFCcyElPTzbqmfRKpykzXbZWV08
S2K7W5CLRnscq1bs7kQ=</ds:SignatureValue>
      <ds:KeyInfo Id="KI-1B492B473942FA437213458421423902">
        <wsse:SecurityTokenReference wsu:Id="STR-1B492B473942FA437213458421423903">
          <ds:X509Data>
            <ds:X509IssuerSerial>
              <ds:X509IssuerName> OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign,
OU=VeriSign International Server CA - Class 3, OU="VeriSign, Inc.", O=VeriSign Trust
Network</ds:X509IssuerName>
              <ds:X509SerialNumber> 128206887485853198247210425948683387 78</ds:X509SerialNumber>
            </ds:X509IssuerSerial>
          </ds:X509Data>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
    <wsu:Timestamp wsu:Id="TS-1">
      <wsu:Created> 2014-04-17T12:25:10.973Z</wsu:Created>
      <wsu:Expires> 2014-04-17T12:30:10.973Z</wsu:Expires>
    </wsu:Timestamp>
  </wsse:Security>
</s:Header>
<s:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="BODY-1">
  …
</s:Body>
```

Element values that will be provided at runtime are Expires, Created, X509SerialNumber, X509IssuerName, SignatureValue, the DigestValue for the Body reference “BODY-1”, and the DigestValue for the Timestamp reference “TS-1”. Any of the Id or wsu:Id values (and their corresponding URI value) can be supplied with a different value as long as they comply with XML Id value rules.

The Created and Expires value must be in UTC format (as in the example above) so that Time zone differences can be resolved correctly. The Expires value is required to be exactly 300 seconds later than the Created timestamp value.

Note that Cannex must be provided with a valid Public Key certificate that matches the X509IssuerName and X509SerialNumber values sent in the SOAP Header.

Enhanced security can be acheived by changing the SignatureMethod and DigestMethod elements to the following pairs:

```
<ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256”/>
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

Or

```
<ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"/>
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
```

Appendix A - Username Token Profile 1.1 Examples

```
user: [Fr3d]
password: [Fl!nst0n3]
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-1">
      <wsse:Username>Fr3d</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">STXysYxJ5Gm3EBYJ0QF3QXQ304U=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">3Q1ygb9JWhYpdJmmRiBWYw==</wsse:Nonce>
      <wsu:Created>2013-01-25T20:42:31.622Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>


user: [B8rn3y]
password: [Rubbl3]
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-2">
      <wsse:Username>B8rn3y</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">EcLlOKrdU4qfV5LY4BfUv87Z34s=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">99NV+9YJf0pgqTUPlYp9+A==</wsse:Nonce>
      <wsu:Created>2013-01-25T20:42:33.230Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>


user: [Cl8rk3]
password: [K3nt]
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-3">
      <wsse:Username>Cl8rk3</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">+DDGXA12ioZPBE5QY2OsjOpr+Ag=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">j0YhLbkLowHTQg/5l/trjQ==</wsse:Nonce>
      <wsu:Created>2013-01-25T20:42:34.745Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>


user: [L01s]
password: [L8n3]
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-4">
      <wsse:Username>L01s</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">kAcAN/hS/OIIBauh+6MqQ8cY388=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">7HJJmc1dLppBRtGExuWd6g==</wsse:Nonce>
      <wsu:Created>2013-01-25T20:42:36.259Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>
```

```
user: [tr8ff!c]
password: [s3rv3r]
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-5">
      <wsse:Username>tr8ff!c</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">AQjCEZrb25OXj2dCowjIfFMDXt4=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">wTAmCL9tmg6KNpeAQOYubw==</wsse:Nonce>
      <wsu:Created>2013-01-25T20:42:37.789Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>
```

```
user: [c0mm0n]
password: [b8ckup]
<s:Header>
  <wsse:Security 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"
s:mustUnderstand="1">
    <wsse:UsernameToken wsu:Id="UsernameToken-6">
      <wsse:Username>c0mm0n</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">EksybYC+Xv/reZGiedJodfT/FTs=</wsse:Password>
      <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">u2i1bBrgUhr4ZK5AiRHA7A==</wsse:Nonce>
      <wsu:Created>2013-01-25T20:42:39.304Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</s:Header>
```

Appendix B - Username Token Profile 1.1 Coding Help

C#

generate datetime format for XML
```
string s = System.DateTime.Now.ToString("o");
```

generate a unique nonce
```
System.Security.Cryptography.RNGCryptoServiceProvider provider = new
System.Security.Cryptography.RNGCryptoServiceProvider();
byte[] nonce = new byte[16];
provider.GetBytes(nonce);
```

converting string to byte array
```
byte[] byteArray = System.Text.Encoding.UTF8.GetBytes(plainText);
```

combine three byte arrays
```
byte[] b4 = new byte[b1.Length + b2.Length + b3.Length];
System.Buffer.BlockCopy(b1, 0, b4, 0, b1.Length);
System.Buffer.BlockCopy(b2, 0, b4, b1.Length, b2.Length);
System.Buffer.BlockCopy(b3, 0, b4, b1.Length + b2.Length, b3.Length);
```

generate SHA1 hash
```
System.Security.Cryptography.SHA1Managed sha1 = new System.Security.Cryptography.SHA1Managed())
byte[] hash = sha1.ComputeHash(byteArray);
```

encode a Base64 string
```
string s = System.Convert.ToBase64String(byteArray);
```

Java

Reference implementation can be found at: http://ws.apache.org/wss4j/index.html

Relevant function from wss4j implementation:

```
public static String doPasswordDigest(String nonce, String created, byte[] password) {
        String passwdDigest = null;
        try {
            byte[] b1 = nonce != null ? Base64.decode(nonce) : new byte[0];
            byte[] b2 = created != null ? created.getBytes("UTF-8") : new byte[0];
            byte[] b3 = password;
            byte[] b4 = new byte[b1.length + b2.length + b3.length];
            int offset = 0;
            System.arraycopy(b1, 0, b4, offset, b1.length);
            offset += b1.length;

            System.arraycopy(b2, 0, b4, offset, b2.length);
            offset += b2.length;

            System.arraycopy(b3, 0, b4, offset, b3.length);

            byte[] digestBytes = WSSecurityUtil.generateDigest(b4);
            passwdDigest = Base64.encode(digestBytes);
        } catch (Exception e) {
            if (DO_DEBUG) {
                LOG.debug(e.getMessage(), e);
            }
        }
        return passwdDigest;
    }
```