eID Security Token Service Specifications

Version 0.1.6-SNAPSHOT

Frank Cornelis

Published Dec 04, 2014

Abstract

This document details on the eID Security Token Service specifications.

1. Introduction

eID authentication in a web context can be easily achieved using for example the eID Identity Provider. Different protocols can be used for web based authentication: SAML 2.0 Browser POST profile, WS-Federation web passive profile, OpenID 2.0, OAuth 2.0 or OpenID Connect. When a relying party (a web application) authenticates an end-user, based on the eID, it can trust the received claims.

In the context of native (desktop/fat-client) applications however, achieving a trust relationship towards a remote relying party can be a challenge. In this document we define the specifications for an Identity Provider Security Token Service (IP-STS) that accepts eID as primary credential. This eID IP-STS is based on open standards like *[WS-Trust]* . The most important features of the eID IP-STS are:

- The eID IP-STS is based on open standards.
- The eID IP-STS offers a secure challenge-response authentication based on the eID card.
- The eID IP-STS can be used with different out-of-the-box STS clients like Windows Identity Foundations or Apache CXF STS Client.
- The eID IP-STS also supports 'lightweight' profiles for clients that have difficulties constructing a *[WS-Security]* based signature.
- The usage of symmetric proof-of-possession keys can be made either mandatory or optional.
- The eID IP-STS provides a mechanism to allow remote relying parties to trust the authenticated user.

1.1. Namespaces

The XML namespaces used in the following sections are described in *Table 1, “XML Namespaces”* .

Table 1. XML Namespaces

Prefix	Namespace
`ds`	`http://www.w3.org/2000/09/xmldsig#`
`soap`	`http://www.w3.org/2003/05/soap-envelope`
`wsa`	`http://www.w3.org/2005/08/addressing`
`wsse`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd`
`wsu`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`
`wst`	`http://docs.oasis-open.org/ws-sx/ws-trust/200512`
`wsp`	`http://schemas.xmlsoap.org/ws/2004/09/policy`
`ec`	`http://www.w3.org/2001/10/xml-exc-c14n#`
`wsc`	`http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512`
`saml2`	`urn:oasis:names:tc:SAML:2.0:assertion`
`xsi`	`http://www.w3.org/2001/XMLSchema-instance`
`fed`	`http://docs.oasis-open.org/wsfed/federation/200706`

Prefix	Namespace
`auth`	`http://docs.oasis-open.org/wsfed/authorization/200706`
`sp`	`http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702`
`wsaw`	`http://www.w3.org/2006/05/addressing/wsdl`
`wsam`	`http://www.w3.org/2007/05/addressing/metadata`
`wsx`	`http://schemas.xmlsoap.org/ws/2004/09/mex`
`wsid`	`http://schemas.xmlsoap.org/ws/2005/05/identity`
`wst14`	`http://docs.oasis-open.org/ws-sx/ws-trust/200802`

1.2. References

[Base64] S. Josefsson, The Base16, Base32, and Base64 Data Encodings , The Internet Society, 2006 http://tools.ietf.org/html/rfc4648 *[http://tools.ietf.org/html/rfc4648]*

[Excl-C14N] J. Boyer et al., Exclusive XML Canonicalization Version 1.0 , World Wide Web Consortium, July 2002 http://www.w3.org/TR/xml-exc-c14n/ *[http://www.w3.org/TR/xml-exc-c14n/]*

[RFC 2616] R. Fielding et al., Hypertext Transfer Protocol - HTTP/1.1. , http://www.ietf.org/rfc/rfc2616.txt IETF (Internet Engineering Task Force) RFC 2616, June 1999.

[SOAP] W3C, SOAP Version 1.2 , SOAP Version 1.2 *[http://www.w3.org/TR/soap12-part1/]* W3C Recommendation 27 April 2007

[RFC 2246] T. Dierks, C. Allen, The TLS Protocol Version 1.0 , http://www.ietf.org/rfc/rfc2246.txt IETF (Internet Engineering Task Force) RFC 2246, January 1999.

[WS-SecConv] A. Nadalin et al., WS-SecureConversation 1.4 , OASIS, February 2009 http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/ws-secureconversation.html *[http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/ws-secureconversation.html]*

[WS-Security] Kelvin Lawrence, Chris Kaler, OASIS Web Services Security: SOAP Message Security 1.1 , OASIS, February 2006 OASIS WS-Security 1.1 *[https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf]*

[WS-SP] Kelvin Lawrence, Chris Kaler, OASIS WS-SecurityPolicy 1.2 , OASIS, July 2007 OASIS WS-SecurityPolicy 1.2 *[http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html]*

[WSS-UT] Anthony Nadalin, Chris Kaler, Ronald Monzillo, Phillip Hallam-Baker OASIS Web Services Security: UsernameToken Profile 1.1 , OASIS, February 2006 OASIS WS-Security UsernameToken Profile 1.1 *[https://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf]*

[WSS-SAML] Anthony Nadalin, Chris Kaler, Ronald Monzillo, Phillip Hallam-Baker OASIS Web Services Security: SAML Token Profile 1.1 , OASIS, June 2005 *OASIS WS-Security SAML Token Profile 1.1 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-pr-SAMLTokenProfile-01.html]*

[WSS-X509] Anthony Nadalin, Chris Kaler, Ronald Monzillo, Phillip Hallam-Baker OASIS Web Services Security: X.509 Certificate Token Profile 1.1 , OASIS, February 2006 *OASIS WS-Security X.509 Certificate Token Profile 1.1 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf]*

[WS-Trust] A. Nadalin et al., WS-Trust 1.3 , OASIS, March 2007 *http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html [http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html]*

[WS-Trust14] A. Nadalin et al., WS-Trust 1.4 , OASIS, 25 April 2012 *http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html [http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html]*

[XHTML] XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) , World Wide Web Consortium Recommendation, August 2002 *http://www.w3.org/TR/xhtml1/ [http://www.w3.org/TR/xhtml1/]*

[XMLSig] D. Eastlake et al., XML-Signature Syntax and Processing , W3C Recommendation, June 2008 *http://www.w3.org/TR/xmldsig-core/*

[XML-ns] T. Bray, D. Hollander, A. Layman, Namespaces in XML , W3C Recommendation, January 1999 *http://www.w3.org/TR/1999/REC-xml-names-19990114 [http://www.w3.org/TR/1999/REC-xml-names-19990114]*

[WSI-BSP] Michael McIntosh, Martin Gudgin, K. Scott Morrison, Abbie Barbir, Basic Security Profile Version 1.1 , The Web Services-Interoperability Organization (WS-I), January 2010 *http://www.ws-i.org/profiles/basicsecurityprofile-1.1.html [http://www.ws-i.org/profiles/basicsecurityprofile-1.1.html]*

[WS-MEX] Doug Davis, Ashok Malhotra, Katy Warr, Wu Chou, Web Services Metadata Exchange (WS-MetadataExchange) , W3C Recommendation 13 December 2011 *http://www.w3.org/TR/ws-metadata-exchange/ [http://www.w3.org/TR/ws-metadata-exchange/]*

2. X.509 Certificate Token Profile

The first authentication construct is based on the OASIS WS-Security X.509 Certificate Profile version 1.1 *[WSS-X509]* .

2.1. Request

The WS-Trust request SOAP message from the native client towards the eID IP-STS looks as follows.

```
<soap:Envelope>
  <soap:Header>
    <wsa:Action soap:mustUnderstand="1">
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
    </wsa:Action>
    <wsa:MessageID>urn:uuid:...</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To s:mustUnderstand="1" u:Id="to">https://...</wsa:To>
    <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>2012-01-17T09:31:27.700Z</wsu:Created>
        <wsu:Expires>2012-01-17T09:36:27.700Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="uuid-bst"
       ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"
         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary">
        ...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <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="#timestamp">
            <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>...</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#to">
            <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>...</DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#uuid-bst"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken>
      <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
      </wst:RequestType>
      <wst:KeyType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer
      </wst:KeyType>
      <wst:TokenType>
       http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
      <wst:TokenType>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>urn:some-target-application</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
    </wst:RequestSecurityToken>
  </soap:Body>
</soap:Envelope>
```

The client sends this request towards the eID IP-STS over SSL (*[RFC 2246]*). This allows the client to authenticate the eID IP-STS and gives us confidentiality and integrity.

The WS-Security XML signature SHOULD at least digest the `<wsa:To>` element and the `<wsu:Timestamp>` element to achieve the required security properties for an entity authentication. The signing of the `<wsa:To>` element prevents reuse of the XML signature within another context (MITM attack). The value of the `<wsa:To>` element SHOULD correspond with the address of the eID IP-STS instance used by the client application. The signing of the `<wsu:Timestamp>` element serves as proof of message freshness towards the eID IP-STS as premisse for the entity authentication.

Ideally `<wsse:BinarySecurityToken>` would be using `#X509PKIPathv1` for `wsse:BinarySecurityToken/@ValueType` as the eID IP-STS needs the complete X509 certificate chain to be able to perform a PKI validation. However, as Windows Communication Foundation only supports `#X509v3` the eID IP-STS must be able to contruct the certificate chain itself.

Note that an eID IP-STS implementation can also accept other non-eID X509 certificates.

To ensure interoperability with Windows Communication Foundation, the eID IP-STS should not require compliance with WS-I Basic Security Profile Version 1.1 (*[WSI-BSP]*).

To ensure interoperability with Apache CXF, the eID IP-STS SHOULD also support the usage of the `<wst:SecondaryParameters>` element.

The `<wst:KeyType>` element indicates which type of proof of possession key should be returned by the eID IP-STS. In this case, the client SHOULD require a bearer token that does not provide any proof of possession.

The `<wst:TokenType>` element indicates the type of the token that should be returned by the eID IP-STS. In this case a SAML 2.0 assertion bearer token will be returned.

The `<wsp:AppliesTo>` elements allows the eID IP-STS to determine which claims it can release within the response SAML assertion.

2.2. Processing

The eID IP-STS performs the following processing:

- The XML signature SHOULD be verified using the `<wsse:BinarySecurityToken>` end-entity certificate.
- The `<wsse:BinarySecurityToken>` X509 certificate chain should be validated using common PKI validation rules (OCSP, CRL).
- The XML signature SHOULD at least sign the `<wsa:To>` element and the `<wsu:Timestamp>` element.
- The `<wsu:Timestamp>` element SHOULD contain a valid time mark.
- The content of the `<wsa:To>` element SHOULD correspond with the eID IP-STS endpoint address.
- The `<wsp:AppliesTo>` SHOULD indicate a known application.

2.3. Response

After a successful validation of the request message the eID IP-STS responds as follows.

```
<soap:Envelope>
  <soap:Header>
    <wsa:Action soap:mustUnderstand="1">
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
    </wsa:Action>
    <wsa:RelatesTo>urn:uuid:...</wsa:RelatesTo>
    <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>2012-01-17T09:31:27.700Z</wsu:Created>
        <wsu:Expires>2012-01-17T09:36:27.700Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse>
```

```
        <wst:Lifetime>
          <wsu:Created>2012-01-17T09:31:29.143Z</wsu:Created>
          <wsu:Expires>2012-01-17T10:31:29.143Z</wsu:Expires>
        </trust:Lifetime>
        <wsp:AppliesTo>
          <wsa:EndpointReference>
            <wsa:Address>urn:some-target-application</a:Address>
          </wsa:EndpointReference>
        </wsp:AppliesTo>
        <wst:RequestedSecurityToken>
          <saml2:Assertion ID="assertionId"
            IssueInstant="2011-05-09T09:36:37.359Z" Version="2.0">
            <saml2:Issuer>https://...</saml2:Issuer>
            <saml2:Subject>
              <saml2:NameID
                Format="urn:oasis:names:tc:SAML:1.1:nameid-format:transient">
                71715100070
              </saml2:NameID>
              <saml2:SubjectConfirmation
                Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
            </saml2:Subject>
            <saml2:Conditions NotBefore="2011-05-09T08:36:37.359Z"
              NotOnOrAfter="2011-05-09T10:36:37.359Z">
              <saml2:AudienceRestriction>
                <saml2:Audience>
                  urn:some-target-application
                </saml2:Audience>
              </saml2:AudienceRestriction>
            </saml2:Conditions>
            <saml2:AuthnStatement AuthnInstant="2011-05-09T09:36:37.515Z">
              <saml2:AuthnContext>
                <saml2:AuthnContextClassRef>
                  urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI
                </saml2:AuthnContextClassRef>
              </saml2:AuthnContext>
            </saml2:AuthnStatement>
            <ds:Signature>
              <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="#assertion">
                  <ds:Transforms>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
                    <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>...</ds:DigestValue>
                </ds:Reference>
              </ds:SignedInfo>
              <ds:SignatureValue>...</ds:SignatureValue>
              <ds:KeyInfo>
                <ds:X509Data>
                  <ds:X509Certificate>...</ds:X509Certificate>
                </ds:X509Data>
              </ds:KeyInfo>
            </ds:Signature>
          </saml2:Assertion>
        </wst:RequestedSecurityToken>
        <wst:RequestedAttachedReference>
          <wsse:SecurityTokenReference
           wsse:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0">
            <wsse:KeyIdentifier
                ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLID">
              assertionId
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </wst:RequestedAttachedReference>
        <wst:TokenType>
       http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
        </wst:TokenType>
        <wst:RequestType>
          http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
        </wst:RequestType>
        <wst:KeyType>
          http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer
        </wst:KeyType>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </soap:Body>
</soap:Envelope>
```

The value of the `<wsa:RelatesTo>` element corresponds with the value of the `<wsa:MessageID>` element within the request message.

Both the `<wsu:Timestamp>` and the `<wst:Lifetime>` serve as proof of the message freshness towards the client.

The `<wst:RequestedSecurityToken>` element contains the `<saml2:Assertion>` SAML 2.0 bearer token. The SAML 2.0 assertion has been signed using the eID IP-STS signing certificate. This XML signature MAY be verified by the requesting client before consuming the `<saml2:NameID>` element within the SAML 2.0 assertion. The XML signature SHOULD be verified by alien relying parties when the SAML 2.0 assertion is used as a bearer token towards WS-Security secured SOAP web services.

2.4. Security Policy

To easy integration with existing STS clients, eID IP-STS implementations should use the following WS-SecurityPolicy *[WS-SP]* configuration.

```
<wsp:Policy wsu:Id="SecurityTokenServicePolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken />
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:IncludeTimestamp />
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:EndorsingSupportingTokens>
        <wsp:Policy>
          <sp:X509Token
    sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssX509V3Token11 />
            </wsp:Policy>
          </sp:X509Token>
          <sp:SignedParts>
            <sp:Header Name="To"
              Namespace="http://www.w3.org/2005/08/addressing" />
          </sp:SignedParts>
        </wsp:Policy>
      </sp:EndorsingSupportingTokens>
      <wsaw:UsingAddressing />
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

To improve interoperability, eID IP-STS implementations should support WS-MetadataExchange *[WS-MEX]* via the `/mex` suffix.

3. SAML Token Profile

A client can used the SAML 2.0 bearer token to secure the communication towards relying parties. A SOAP request message can embed the SAML 2.0 bearer token according to OASIS WS-Security SAML Token Profile 1.1.1 *[WSS-SAML]* . The following SOAP request message demonstrates the usage of such a SAML 2.0 bearer token.

```
<soap:Envelope>
  <soap:Header>
    <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>2012-01-17T09:31:27.700Z</wsu:Created>
        <wsu:Expires>2012-01-17T09:36:27.700Z</wsu:Expires>
      </wsu:Timestamp>
      <saml2:Assertion ID="assertion"
            IssueInstant="2011-05-09T09:36:37.359Z" Version="2.0">
        ...
      </saml2:Assertion>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

A relying party web service supporting WS-SecurityPolicy *[WS-SP]* could use the following security policy.

```
<wsp:Policy wsu:Id="ExampleSamlSecurityPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken />
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:IncludeTimestamp />
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:SamlToken>
            <wsp:Policy>
              <sp:WssSamlV20Token11 />
            </wsp:Policy>
          </sp:SamlToken>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <sp:Wss11>
        <wsp:Policy>
          <sp:MustSupportRefEmbeddedToken />
        </wsp:Policy>
      </sp:Wss11>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

A relying party web service could even refer directly to the eID IP-STS within the security policy as shown below.

```
<wsp:Policy wsu:Id="ExampleSamlSecurityPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken />
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:IncludeTimestamp />
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:IssuedToken
                          sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
```

```
            <sp:RequestSecurityTokenTemplate>
              <wst:TokenType>
                   http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0
              </wst:TokenType>
              <wst:KeyType>
                http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer
              </wst:KeyType>
            </sp:RequestSecurityTokenTemplate>
            <wsp:Policy>
              <sp:RequireInternalReference />
            </wsp:Policy>
            <sp:Issuer>
              <wsa:Address>
                http://www.w3.org/2005/08/addressing/anonymous
              </wsa:Address>
              <wsa:Metadata>
                <wsx:Metadata>
                  <wsx:MetadataSection>
                    <wsx:MetadataReference>
                      <wsa:Address>
                        https://www.e-contract.be/iam/sts/mex
                      </wsa:Address>
                    </wsx:MetadataReference>
                  </wsx:MetadataSection>
                </wsx:Metadata>
              </wsa:Metadata>
            </sp:Issuer>
          </sp:IssuedToken>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <wsam:Addressing wsp:Optional="false">
        <wsp:Policy />
      </wsam:Addressing>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

The above security policy uses the WS-MetadataExchange feature of the eID IP-STS to allow for easy configuration of STS clients.

4. Validating SAML assertions

When a relying party receives a SAML 2.0 assertion as bearer token it SHOULD validate the SAML assertion before proceeding. The relying party should at least verify the following conditions:

- Validate the XML signature on the SAML assertion. The XML signature should contain a `<ds:Reference>` referring to the top-level `<saml2:Assertion>` element.

- The relying party SHOULD verify whether the value of the `<saml2:Issuer>` element corresponds with the trusted eID IP-STS instance.
- The relying party SHOULD verify whether the SAML signatory corresponds with the trusted eID IP-STS instance.
- The relying party SHOULD verify the freshness of the SAML 2.0 assertion via the `NotBefore` and `NotAfter` attributes of the `<saml2:Conditions>` element.
- The relying party SHOULD verify that the SAML 2.0 assertion is indeed intended for him via the `<saml2:Audience>` element.

A relying party can setup the trust relationship towards the eID IP-STS instance in different ways. The relying party could for example keep reference of the SHA1 fingerprint of the trusted eID IP-STS signing certificate. The relying party could also keep reference of the X509 certificate of the trusted eID IP-STS instance.

To aid the relying party in the bootstrapping of the trust relationship towards the eID IP-STS, an eID IP-STS instance should expose an OASIS SAML 2.0 metadata document. This SAML 2.0 metadata document SHOULD at least contain the following elements.

```
<md:EntityDescriptor entityID="https://...">
  <md:RoleDescriptor xsi:type="fed:SecurityTokenServiceType"
                  protocolSupportEnumeration="http://docs.oasis-open.org/wsfed/
federation/200706">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo>
        <ds:X509Data>
          <ds:X509Certificate>...</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <fed:TokenTypesOffered>
      <fed:TokenType Uri="urn:oasis:names:tc:SAML:2.0" />
    </fed:TokenTypesOffered>
    <fed:SecurityTokenServiceEndpoint>
      <wsa:EndpointReference>
        <wsa:Address>https://...</wsa:Address>
      </wsa:EndpointReference>
    </fed:SecurityTokenServiceEndpoint>
  </md:RoleDescriptor>
</md:EntityDescriptor>
```

A relying party SHOULD retrieve this SAML 2.0 metadata document over SSL. The eID IP-STS signing certificate is located within the `<md:KeyDescriptor>` element.

5. STS based Validation

While a relying party can perfectly validate a received SAML assertion itself as described under *Section 4, “Validating SAML assertions”* , it might be easier for some relying parties to use the eID IP-STS itself to validate the SAML assertions.

5.1. Request

A relying party can send the following STS validation request to the eID IP-STS.

```
<soap:Envelope>
  <soap:Header>
    <wsa:Action soap:mustUnderstand="1">
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Validate
    </wsa:Action>
    <wsa:MessageID>urn:uuid:...</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To s:mustUnderstand="1" u:Id="to">https://...</wsa:To>
    <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>2012-01-17T09:31:27.700Z</wsu:Created>
        <wsu:Expires>2012-01-17T09:36:27.700Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken>
      <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Validate
      </wst:RequestType>
      <wst:TokenType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Status
      </wst:TokenType>
      <wst:ValidateTarget>
        <saml2:Assertion ID="assertionId"
          IssueInstant="2012-02-09T09:27:17.433Z" Version="2.0">
          ...
        </saml2:Assertion>
      </wst:ValidateTarget>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>urn:some-target-application</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
    </wst:RequestSecurityToken>
  </soap:Body>
```

```
</soap:Envelope>
```

The relying party should send the request over SSL towards the trusted eID IP-STS instance.

Via the `<wsp:AppliesTo>` element the relying party can restrict the application context in which the SAML assertion should be validated. Since according to the WS-Trust specification *[WS-Trust]* the `<wsp:AppliesTo>` element is not mandatory, eID IP-STS implementations should also consider this element as optional. This to prevent that vanilla STS clients cannot invoke the validation operation on the STS.

5.2. Processing

The eID IP-STS SHOULD at least perform the following processing:

- The `<wsu:Timestamp>` element SHOULD contain a valid time mark.
- The content of the `<wsa:To>` element SHOULD correspond with the eID IP-STS endpoint address.
- Validate the XML signature on the SAML assertion. The XML signature should contain a `<ds:Reference>` referring to the top-level `<saml2:Assertion>` element.
- The eID IP-STS SHOULD verify whether the value of the `<saml2:Issuer>` element equals its own name.
- The eID IP-STS SHOULD verify that he is the SAML signatory.
- The eID IP-STS SHOULD verify the freshness of the SAML 2.0 assertion via the `NotBefore` and `NotAfter` attributes of the `<saml2:Conditions>` element.
- The eID IP-STS SHOULD verify that the SAML 2.0 assertion `<saml2:Audience>` element value corresponds with the `<wsp:AppliesTo>` element value from the request. This only if the `<wsp:AppliesTo>` element has been provided.

5.3. Response

Upon successful processing, the eID IP-STS responds with:

```
<soap:Envelope>
  <soap:Header>
    <wsa:Action soap:mustUnderstand="1">
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/ValidateFinal
    </wsa:Action>
    <wsa:RelatesTo>urn:uuid:...</wsa:RelatesTo>
    <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp>
```

```
        <wsu:Created>2012-02-09T09:27:19.197Z</wsu:Created>
        <wsu:Expires>2012-02-09T09:32:19.197Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Status
      </wst:TokenType>
      <wst:Status>
        <wst:Code>
          http://docs.oasis-open.org/ws-sx/ws-trust/200512/status/valid
        </wst:Code>
        <wst:Reason>
          The Trust service successfully validated the input
        </wst:Reason>
      </wst:Status>
    </wst:RequestSecurityTokenResponse>
  </soap:Body>
</soap:Envelope>
```

The relying party SHOULD at least check whether the `<wst:Code>` element value indeed indicates a `valid` status.

6. Attributes

The active authentication as specified under *Section 2, “X.509 Certificate Token Profile”* results in a SAML 2.0 assertion that only contains a subject `<saml2:NameID>` as unique user identifier. This section describes various options to enrich the retrieved SAML assertion with attributes. Attributes can have different origins:

- Attributes can be self claimed.
- Attributes can be derived from the eID card. As the eID card also offers different attributes (name, address, ...), relying parties probably also want to access these eID attributes.

To allow the eID IP-STS to issue SAML assertions containing attributes, the relying party needs to pass a specific SAML 2.0 assertion as part of the *Section 2.1, “Request”* . This specific SAML assertion should be passed as SAML 2.0 bearer token within the `<wst14:ActAs>` WS-Trust 1.4 *[WS-Trust14]* element. We are using `<wst14:ActAs>` to achieve maximum interoperability when using web service frameworks like Apache CXF or Windows Communication Foundation. The specific SAML assertion looks as follows.

```
<saml2:Assertion ID="assertion-id"
  IssueInstant="2011-05-09T09:36:37.359Z" Version="2.0">
  <saml2:Issuer>self-claimed</saml2:Issuer>
  <saml2:AttributeStatement>
    <saml2:Attribute Name="urn:be:e-contract:sts:eid:identity">
      <saml2:AttributeValue xsi:type="xs:base64Binary">
        ...
      </saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name="urn:be:e-contract:sts:eid:identity-signature">
      <saml2:AttributeValue xsi:type="xs:base64Binary">
        ...
      </saml2:AttributeValue>
    </saml2:Attribute>
     <saml2:Attribute Name="urn:be:e-contract:iam:claims:self-claimed:software-
key">
      <saml2:AttributeValue xsi:type="xs:string">
        ...
      </saml2:AttributeValue>
    </saml2:Attribute>
  </saml2:AttributeStatement>
</saml2:Assertion>
```

Depending on the attributes that the eID IP-STS should return, other `<saml2:Attribute>` elements should be passed within the specific SAML 2.0 assertion. The eID IP-STS should support the following specific input attributes:

`urn:be:e-contract:sts:eid:identity`
: The eID identity file. The attribute value type should be `xs:base64Binary` .

`urn:be:e-contract:sts:eid:identity-signature`
: The eID identity file signature by the national registration authority. The attribute value type should be `xs:base64Binary` .

`urn:be:e-contract:sts:eid:nr-cert`
: The X509 signing certificate used by the national registration authority. The attribute value type should be `xs:base64Binary` .

`urn:be:e-contract:sts:eid:address`
: The eID address file. The attribute value type should be `xs:base64Binary` .

`urn:be:e-contract:sts:eid:address-signature`
: The eID address file signature by the national registration authority. The attribute value type should be `xs:base64Binary` .

`urn:be:e-contract:sts:eid:photo`
: The eID photo file. The attribute value type should be `xs:base64Binary` .

`urn:be:e-contract:iam:claims:self-claimed:office-key`
: The self claimed office key. The attribute value type should be `xs:string` .

`urn:be:e-contract:iam:claims:self-claimed:software-key`
: The self claimed software key. The attribute value type should be `xs:string` .

Eventually the eID IP-STS will deliver attributes as part of the SAML 2.0 bearer token. These attributes are made available as `<saml2:AttributeStatement>` attributes within the SAML 2.0 bearer token. The eID IP-STS SHOULD support the following attributes:

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name`
: The URI for a claim that specifies the full name of an eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname`
: The URI for a claim that specifies the surname of an eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname`
: The URI for a claim that specifies the given name of an eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/x500distinguishedname`
: The URI for a distinguished name claim of the eID X.509 authentication certificate. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country`
: The URI for a claim that specifies the given name of an eID user. For eID users this defaults to BE. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/dateofbirth`
: The URI for a claim that specifies the date of birth of an eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/gender`
: The URI for a claim that specifies the gender an eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/locality`
: The URI for a claim that specifies the locality in which an eID user resides. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/postalcode`
: The URI for a claim that specifies the postal code of an eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier`
: The URI for a claim that specifies the private personal identifier (PPI) of the eID user. The attribute value type should be `xs:string` .

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/streetaddress`
: The URI for a claim that specifies the street address (street and house number) of the eID user. The attribute value type should be `xs:string` .

`urn:be:e-contract:iam:claims:self-claimed:office-key`
: The self claimed office key. The attribute value type should be `xs:string` .

`urn:be:e-contract:iam:claims:self-claimed:software-key`
: The self claimed software key. The attribute value type should be `xs:string` .

Depending on the eID IP-STS policy, the eID IP-STS will enrich the SAML 2.0 bearer token with attributes based on:

- The `<wsp:AppliesTo>` element.
- The optional `<wst:Claims>` element that should be part of the `<wst:RequestSecurityToken>` element.

6.1. Request

A request that demonstrates the usage of eID attributes is given below. Here we omitted most elements from the *Section 2.1, “Request”* message to better highlight the attribute sections.

```
<soap:Envelope>
  <soap:Header>
    ...
    <wsse:Security soap:mustUnderstand="1">
      ...
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken>
      ...
      <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
        <wsid:ClaimType
          Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
        <wsid:ClaimType
        Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
      </wst:Claims>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>urn:some-target-application</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst14:ActAs>
        <saml2:Assertion ID="assertion-id"
          IssueInstant="2011-05-09T09:36:37.359Z" Version="2.0">
          <saml2:AttributeStatement>
```

```
              <saml2:Attribute Name="urn:be:e-contract:sts:eid:identity">
                <saml2:AttributeValue xsi:type="xs:base64Binary">
                  ...
                </saml2:AttributeValue>
              </saml2:Attribute>
            <saml2:Attribute Name="urn:be:e-contract:sts:eid:identity-signature">
                <saml2:AttributeValue xsi:type="xs:base64Binary">
                ...
                </saml2:AttributeValue>
              </saml2:Attribute>
              <saml2:Attribute Name="urn:be:e-contract:sts:eid:nr-cert">
                <saml2:AttributeValue xsi:type="xs:base64Binary">
                  ...
                </saml2:AttributeValue>
              </saml2:Attribute>
            </saml2:AttributeStatement>
          </saml2:Assertion>
        </wst14:ActAs>
      </wst:RequestSecurityToken>
    </soap:Body>
  </soap:Envelope>
```

As you can see from the example, the request for the `surname` and `givenname` claims, will require the provisioning of the eID identity file, the eID identity file signature, and the eID national registration signing certificate. Otherwise the eID IP-STS cannot verify the security properties on these eID data files, and thus cannot securely parse and assert the requested eID attributes. The following table *Table 2, “Attribute mapping”* summarizes which input attributes should be provided per requested claim type. Note that we omitted the prefixes for brevity.

Table 2. Attribute mapping

ClaimType Uri	Attribute Name Uri
`name`	`identity` , `identity-signature` , `nr-cert`
`surname`	`identity` , `identity-signature` , `nr-cert`
`givenname`	`identity` , `identity-signature` , `nr-cert`
`x500distinguishedname`	Nothing.
`dateofbirth`	`identity` , `identity-signature` , `nr-cert`
`gender`	`identity` , `identity-signature` , `nr-cert`
`locality`	`identity` , `identity-signature` , `nr-cert` , `address` , `address-signature`
`postalcode`	`identity` , `identity-signature` , `nr-cert` , `address` , `address-signature`
`streetaddress`	`identity` , `identity-signature` , `nr-cert` , `address` , `address-signature`

ClaimType Uri	Attribute Name Uri
`software-key`	`software-key`
`office-key`	`office-key`

6.2. Processing

When the eID IP-STS receives eID data files via the additional `<saml2:Assertion>` within the request, is SHOULD perform the following additional processing, next to the processing already defined under *Section 2.2, “Processing”* .

- Depending on the configured policy, determine whether the requested `<wst:Claims>` are permitted by the calling application as indicated via `<wsp:AppliesTo>` .
- Check whether the provided eID data files are sufficient to meet the requested claims.
- Validate the eID identity file signature using the received national registration X509 certificate.
- Perform a PKI validation against the national registration X509 certificate.
- Check the indicated validity period within the eID identity file.
- Check whether the national registration number within the eID identity file corresponds with the SERIALNUMBER field of the eID authentication certificate distinguished name.
- Parse and (depending on the claims type) transform the eID identity attributes towards the requested claims.

Similar processing applies to the eID address file if claims related to this file are requested.

6.3. Response

After successful processing, the eID IP-STS responds as follows. Again we omitted several response elements to highlight the eID attribute relevant elements.

```
<soap:Envelope>
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse>
        ...
        <wst:RequestedSecurityToken>
          <saml2:Assertion ID="assertionId"
            IssueInstant="2011-05-09T09:36:37.359Z" Version="2.0">
            ...
            <saml2:AttributeStatement>
```

```
              <saml2:Attribute
                  Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
surname">
                <saml2:AttributeValue xsi:type="xs:string">
                  Cornelis
                </saml2:AttributeValue>
              </saml2:Attribute>
              <saml2:Attribute
                  Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
givenname">
                <saml2:AttributeValue xsi:type="xs:string">
                  Frank
                </saml2:AttributeValue>
              </saml2:Attribute>
            </saml2:AttributeStatement>
          </saml2:Assertion>
        </wst:RequestedSecurityToken>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </soap:Body>
</soap:Envelope>
```

6.4. Security Policy

A relying party web service could use the following security policy.

```
<wsp:Policy wsu:Id="ExampleSamlSecurityPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken />
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:IncludeTimestamp />
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:IssuedToken
```

```
                          sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
            <sp:RequestSecurityTokenTemplate>
              <wst:TokenType>
                   http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0
              </wst:TokenType>
              <wst:KeyType>
                http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer
              </wst:KeyType>
          <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
                <wsid:ClaimType
                         Uri="http://schemas.microsoft.com/ws/2008/06/identity/
claims/role" />
                ...
              </wst:Claims>
            </sp:RequestSecurityTokenTemplate>
            <wsp:Policy>
              <sp:RequireInternalReference />
            </wsp:Policy>
            <sp:Issuer>
              <wsa:Address>
                http://www.w3.org/2005/08/addressing/anonymous
              </wsa:Address>
              <wsa:Metadata>
                <wsx:Metadata>
                  <wsx:MetadataSection>
                    <wsx:MetadataReference>
                      <wsa:Address>
                        https://www.e-contract.be/iam/sts/mex
                      </wsa:Address>
                    </wsx:MetadataReference>
                  </wsx:MetadataSection>
                </wsx:Metadata>
              </wsa:Metadata>
            </sp:Issuer>
          </sp:IssuedToken>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <wsam:Addressing wsp:Optional="false">
        <wsp:Policy />
      </wsam:Addressing>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

The above security policy explicitly states the requested claims towards the eID IP-STS via the `<wst:Claims>` element.

7. Holder-of-key tokens

Under *Section 2, “X.509 Certificate Token Profile”* we described a way to retrieve bearer tokens. Bearer tokens do not have proof keys while holder-of-key tokens do. A holder-of-key token is used to ensure that messages sent from subject to RP are indeed from the subject claiming to have sent them. This protocol-level protection guards against man in the middle attacks. Depending on the relying party's web service policy some relying parties might require holder-of-key tokens.

In this section we describe how the eID IP-STS can issue holder-of-key tokens where the proof-of-possession key is the eID authentication private key itself. For a relying party to request a holder-of-key token it should extend the *Section 2.1, “Request”* message under `<wst:RequestSecurityToken>` as follows:

```
<wst:KeyType>
  http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey
</wst:KeyType>
<wst:UseKey>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#uuid-bst"/>
  </wsse:SecurityTokenReference>
</wst:UseKey>
```

Here the `<wsse:SecurityTokenReference>` simply refers to the `<wsse:BinarySecurityToken>` that holds the eID authentication X509 certificate chain and which is already part of the `<wsse:Security>` SOAP header. Of course the client proves possession of this key by means of the WS-Security XML signature.

When the eID IP-STS receives a request to issue a holder-of-key token, it will put the following `<saml2:SubjectConfirmation>` within the returned `<saml2:Assertion>` .

```
<saml2:SubjectConfirmation
  Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
  <saml2:SubjectConfirmationData>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>...</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </saml2:SubjectConfirmationData>
</saml2:SubjectConfirmation>
```

By using such eID based holder-of-key token we basically switch from the X509 certificate trust model to a SAML based trust model. Important to note here that the client keeps using the eID authentication key towards relying parties when presenting the corresponding SAML holder-of-key token. The advantage of this mechanism is that the relying party no longer needs to perform a full PKI validation of the eID authentication certificate chain. The relying party can simply trust the eID IP-STS to already have executed the relevant PKI verifications as part of the token issuing. The holder-of-key SAML token is used as described under *Section 3, " SAML Token Profile "* except that we also include an XML signature as part of the WS-Security header. This XML signature refers to the SAML token within its `<ds:KeyInfo>` as follows.

```
<ds:KeyInfo>
  <wsse:SecurityTokenReference
           wsse:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0">
    <wsse:KeyIdentifier
                ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLID">
      assertionId
    </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

A. eID Security Token Service Specifications License

B. eID Security Token Service Project License