>

OIO WS-Trust Profile

Version 1.0.2

IT- & Telestyrelsen
January 2015

Content >

Document History

Date	Version	Initials	Changes
09-06-2009	0.99	SPN	Document ready for OIO public hearing
08-09-2009	1.0	TG	Document updated after public hearing (only editorial changes).
14-12-2009	1.0.1	TG	Updated to clarify that the <ActAs> element is defined under WS-Trust 1.4 namespace whereas all other WS-Trust elements are defined in the 1.3 namespace.
14-06-2010	1.0.2	TG	Added profiling of the <UseKey> element.

Introduction

Identity-based web services are expected to play an important role within Danish eGovernment since they allow IT systems to be connected in a secure, privacy-respecting and interoperable manner.

The needs and requirements of the Danish public sector have been documented in a number of scenarios [Scenarios]. One important component that has been identified is a Security Token Service (STS) that can issue and exchange security tokens in the form of SAML 2.0 Assertions. This document specifies a WS-Trust 1.3 interoperability profile defining the messaging interface of such a security token service. The profile limits the large number choices and flexibility present in WS-Trust and tailors it to the specific context. The goal is to promote interoperability and reduce complexity for implementers.

This profile is meant to be used in two major scenarios: the first is a web site (e.g. a portal) who needs to invoke an identity-based web service at a web service provider (WSP) on behalf of a (browser) user. In order to invoke the web service, the requester must first obtain an identity token representing the user from an STS, and subsequently place it in the web service call. Another important use of the profile is scenarios with "rich clients" that needs to invoke external web services on the user's behalf and therefore also need to obtain security tokens to gain access.

Related profiles

This profile is designed such that all messages conforming to it will be a (true) subset of WS-Trust [WST] 1.3 conforming messages. The XML name space prefixes used in this profile (e.g. `wsu`, `ds`, `wst`, `wsa`, `wsp`) have the same value and meaning as in WS-Trust 1.3 (not repeated here). The only exception is the use of the `<wst14:ActAs>` element which is borrowed from WS-Trust 1.4 since there is no element in WS-Trust 1.3 with the required semantics.

A number of other documents are closely related:

- The [Scenarios] document describes the overall business goals and requirements within Danish eGovernment and shows how the different OIO profiles are combined to achieve these.

The reader is assumed to be familiar with WS-Trust [WST] and SAML.

General Requirements

This chapter contains a number of requirements shared by all WS-Trust 1.3 operations defined in this profile.

The OIO WS-Trust profile does not mandate any specific transport binding or security mechanisms for transmitting the WS-Trust messages. Separate deployment profiles (e.g. [OIO-WST-DEP]) will have to define mappings to e.g. SOAP and further specify security details relevant for the transport binding. However, this profile states high-level requirements for signing message elements below.

<wst:RequestSecurityToken> Usage

- The `<wst:RequestSecurityToken>` element MUST be signed by the requester.
- The requester MAY send security tokens (including certificates) vouching for its identity and signing key as appropriate to the selected binding.
- The requester MAY send security tokens vouching for the identity of a user that is acted on behalf of.

Processing Rules

The STS receiving a WS-Trust message MUST do the following:

- Validate the signature, any certificates, any security tokens and timestamp of the request according to local policy.

<wst:RequestSecurityTokenResponse> Usage

- The response MUST consist of exactly one `<RequestSecurityTokenResponseCollection>` element with exactly one `<RequestSecurityTokenResponse>` element.
- The response MUST be signed by the STS.

Processing Rules

The receiver MUST do the following:

- Validate the signature and timestamp of the response according to local policy.

Issuance Binding

This chapter specifies a number of message requirements in context of the Issuance binding defined in WS-Trust [WST]. Note that the general requirements in the previous chapter MUST also be followed.

Unless described otherwise all message elements and processing rules defined in WS-Trust apply.

<RequestSecurityToken> Requirements

- The `<wst:RequestType>` element MUST indicate the issuance binding and therefore use the following URI: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue`.
- When a service is requesting a token on behalf of a user, an `<wst14:ActAs>` element defined in WS-Trust 1.4[1] MUST be present and include a bootstrap token representing the user. The token SHOULD be embedded directly and thus not be a token reference. The format of the bootstrap token is not specified in this profile as it can vary with different deployments. The OIO bootstrap token profile contains a profile for the web SSO scenario where the user has a browser session with a SAML-based Identity Provider.
- The `<wsp:AppliesTo>` element SHOULD contain an `<wsa:EndpointReference>` identifying the web service provider or specific service for which the identity token should be issued.
- The `<wst:Claims>` element SHOULD include a list of the claims requested to be included in the issued token. The individual claims SHOULD be specified using the `ic:ClaimType` element defined in [ISIP]. Furthermore, the `Dialect` attribute on `wst:Claims` SHOULD be set to `http://schemas.xmlsoap.org/ws/2005/05/identity`
- The `<wst:LifeTime>` MAY be used to indicate the desired valid time range of the token to be issued. The issuer is not obligated to honor this range and MAY return a token with a shorter life time.
- If the requester wants the STS to issue a SAML 2.0 holder-of-key token referencing the requester's X.509 certificate[2], it MUST include a `<wst:UseKey>` with the certificate embedded in a `<wsse:BinarySecurityToken>` element. Here the `ValueType` attribute MUST be set to `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3` and the `EncodingType` attribute set to `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary`

Processing Rules

- The token lifetime SHOULD be set according to local policy and MAY override the lifetime indicated in the request.
- The lifetime set in the response SHOULD match the lifetime in the issued token.
- If the issued token is a SAML 2.0 assertion with a "holder-of-key" subject confirmation element:
    - Any (asymmetric) public key referenced in this element SHOULD match the key used to verify the requester's signature on the request message.

---

[1] The reader should notice that the WS-Trust 1.4 specification by mistake defined the ActAs element in WS-Trust 1.3 namespace (`http://docs.oasis-open.org/ws-sx/ws-trust/200512` denoted "`wst`" here), where it should have been declared in WS-Trust 1.4 namespace (`http://docs.oasis-open.org/ws-sx/ws-trust/200802` denoted "`wst14`" here).

[2] I.e. the existing private key becomes proof-of-possession key for the new token.

    - Any symmetric keys returned MUST be encrypted under the recipient's key (the audience of the token). Further, the generated symmetric key MUST also be returned in a `<wst:RequestedProofToken>` for the requester.
- The token issuer MAY perform an authorization decision before issuing the identity token. Such decisions are out of scope for this profile.
- If the issued token includes an audience restriction (e.g. as in SAML 2), this element MUST be consistent with the `<wsp:AppliesTo>` element in the response. Note that the SAML entity ID in the issued token may be syntactically different from a provider specified in the end point reference in the `<wsp:AppliesTo>` element, but they MUST refer to the same logical entity.
- The token issuer MAY define local policies stating additional conditions for a token to be issued - including requiring the user to have an active SSO session with an Identity Provider. How the token issuer performs such checks is considered private and therefore outside the scope of this profile[3].

Delegation

Delegation of tokens SHOULD NOT be used in the OIO profile.

[3] For example, a bootstrap assertion could include a reference to the current Identity Provider session for the user (e.g. SessionIndex).

Request example (non-normative)

Below is shown a SOAP message with a sample request (some fields have been omitted to keep the example short). Note that the usage of SOAP, WS-Addressing and WS-Security is not normative and will depend on the specific binding and security mechanisms relevant for the deployment:

```
<S11:Envelope xmlns:S11="..." xmlns:wsu="..." xmlns:wsse="..."
        xmlns:xenc="..." xmlns:wst="...">

    <S11:Header>
        <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-
                  trust/200512/RST/Issue</wsa:Action>
        <wsa:MessageID>urn:uuid:99999999-0000-0000….</wsa:MessageID>
        <wsa:To>http://sts.example.org</wsa:To>

        <wsse:Security mustUnderstand="1">
           <wsu:Timestamp wsu:ID="ts"> … </wsu:Timestamp>

           <ds:Signature xmlns:ds="...">
              <ds:SignedInfo> …
                 <ds:Reference URI="#req"> …  </ds:Reference>
                 <ds:Reference URI="#ts"> ... </ds:Reference>
                 <!-- More references to other header elements -->
              </ds:SignedInfo>
              <ds:SignatureValue> … </ds:SignatureValue>
              <ds:KeyInfo>
                <ds:X509Data> … sender certificate … </ds:X509Data>
              </ds:KeyInfo>
           </ds:Signature>
        </wsse:Security>

    </S11:Header>

    <S11:Body wsu:Id="req">
       <wst:RequestSecurityToken Context="urn:uuid:00000…">
          <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:UseKey>
              <wsse:BinarySecurityToken
                 ValueType="http://docs.oasis-open.org/....X509v3"
                 EncodingType="http://docs.oasis-open.org/...Base64Binary">

                     BASE64 encoded value

              </wsse:BinarySecurityToken>
          </wst:UseKey>


          <wst14:ActAs>
            <!-- Include token for user that is acted on behalf of here -->
          </wst14:ActAs>

          <wsp:AppliesTo>
             <wsa:EndpointReference>
                <wsa:Address>http://agency_x.dk</wsa:Address>
             </wsa:EndpointReference>
          </wsp:AppliesTo>

          <wst:Claims
             wst:Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
            <ic:ClaimType
               Uri="http://.../ws/2005/05/identity/claims/givenname"/>
            <ic:ClaimType
               Uri="http://.../ws/2005/05/identity/claims/surname"/>
```

In the above example, the sender signs the request and includes his X.509 certificate to authenticate the request, ensure integrity and establish trust in the signing key. Further, a token representing the user that is acted on behalf of is also included.

<RequestSecurityTokenResponse> requirements

- The `<wst:RequestedSecurityToken>` MUST contain the issued token directly (i.e. not a `SecurityTokenReference`).
- If a symmetric key is returned, a `<wst:RequestedProofToken>` element MUST be returned specifying the generated key or entropy that can be used to compute the key.
- The `<wsp:AppliesTo>` element SHOULD contain the `<wsa:EndpointReference>` from the request. If the token issuer performs authorization decisions, the scope of the resource MAY be overridden.
- The `wst:RequestedAttachedReference` and `wst:RequestedUnattachedReference` elements MUST be included in the response. These allow the requesting Web Service Consumer to use the issued token in messages towards a Web Service Provider (including signing the token via the message signature) without knowing the token type or parsing it.

Response example (non-normative)

Below is shown a SOAP message with a sample response. As before the usage of SOAP, WS-Addressing and WS-Security is not normative and will vary with binding:

```
<S11:Envelope xmlns:S11="..." xmlns:wsu="..." xmlns:wsse="..."
        xmlns:xenc="..." xmlns:wst="...">

    <S11:Header>

        <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-
                  trust/200512/RST/Issue</wsa:Action>
        <wsa:MessageID>urn:uuid:99999777-0000….</wsa:MessageID>
        <wsse:Security mustUnderstand=”1”>
           <wsu:Timestamp wsu:ID=”ts”> … </wsu:Timestamp>
           <ds:Signature xmlns:ds="...">
              <ds:SignedInfo> …
                 <ds:Reference URI="#resp"> …  </ds:Reference>
                 <ds:Reference URI=”#ts”> ... </ds:Reference>
                 <!-- More references to other header elements -->
              </ds:SignedInfo>
              <ds:SignatureValue> … </ds:SignatureValue>
              <ds:KeyInfo>
                 <ds:X509Data> .. STS certificate … </ds:X509Data>
              </ds:KeyInfo>
           </ds:Signature>
        </wsse:Security>
    </S11:Header>
    <S11:Body wsu:Id="resp">
       <wst:RequestSecurityTokenResponseCollection>
          <wst:RequestSecurityTokenResponse Context=”urn:uuid:00000…”>
             <wst:TokenType>
                http://docs.oasis-open.org/wss/oasis-wss-saml-token-
                     profile-1.1#SAMLV2.0
             </wst:TokenType>
             <wst:RequestedSecurityToken>
                <!—- Here comes the issued token -->
                   <saml:Assertion ID="_1234 " Version="2.0" …> …
                   </saml:Assertion>
             </wst:RequestedSecurityToken>
             <wst:RequestedAttachedReference>
                <wsse:SecurityTokenReference …>
                   <wsse:KeyIdentifier ValueType="http://docs.oasis-
                      open.org/wss/oasis-wss-saml-token-profile-
                      1.1#SAMLID>_1234
                   </wsse:KeyIdentifier>
                </wsse:SecurityTokenReference>
             </wst:RequestedAttachedReference>
             <wst:RequestedUnattachedReference>
                <!—- As the attached reference above  -->
             </wst:RequestedUnattachedReference>
             <wsp:AppliesTo>
                <wsa:EndpointReference>
                   <wsa:Address>http://agency_x.dk</wsa:Address>
                </wsa:EndpointReference>
             </wsp:AppliesTo>
             <wst:Lifetime> … </Lifetime>
          </wst:RequestSecurityTokenResponse>
       </wst:RequestSecurityTokenResponseCollection>
    </S11:Body>
</S11:Envelope>
```

The STS has signed the response message and issued a SAML 2.0 assertion with the requested claims about the user. The issued assertion could include a SubjectConfirmation element with a holder-of-key reference to the requester's key.

Security Requirements

This profile simply requires request and response message to be signed to ensure authentication and integrity. Security details including how to map to specific security mechanisms present in bindings are left to deployment profiles.

Architectural Decisions (informational)

This chapter contains a number of architectural decisions which provide the rationale behind important choices made in the profile.

Claims in request

Problem	Should the Web Service Consumer know / discover which claims the Web Service Provider needs in the token to grant access to the requested service (as in <AppliesTo>) and put them in the RST/Claims element for the STS?!
Assumptions	The WSP performs authorization decisions before service access is granted based on the identity of the WSC and the claims in the token. It has formulated an explicit policy specifying the required claims in tokens.
Alternatives	1) The WSC could have the responsibility of determining the claims and specifying them in the RST. 2) The STS could have the responsibility of knowing the WSP's access policy and thus including the right claims in the issued tokens. 3) A combination of the above. 4) Require WSC and WSP to use WS-SecurityPolicy and MetaDataExchange for maximum flexibility.
Analysis	The most general and flexible approach is to specify the claims in the RST message. This does not automatically imply that the WSC has to understand WS-SecurityPolicy and/or dynamically fetch the WSP's policy. When trust relations and claims are fairly static, a WSC may still be statically configured with the required claims and can later evolve to a more dynamic, policy-driven approach. Specifying the claims seems to be more interoperable (see e,g, [ISIP])..
Decision	Recommend the WSC to specify the claims.

Token Format

Problem	Should the profile specify the format of tokens being issued?!
Assumptions	In Danish eGovernment scenarios only SAML 2.0 assertions are relevant.
Alternatives	1) Restrict profile to a SAML 2.0 token profile. 2) Let token profiles be unspecified in this profile.
Analysis	WS-Trust allows the issued tokens to be opaque to the Web Service Consumer. Thus, there is no real reason to restrict the token format. Leaving the token format open will allow use of the profile in other contexts than Danish eGovernment. Instead, it can be left to deployment guides / policies to state, that parties only are required to support SAML 2.0 tokens to allow simple implementations.
Decision	Don't specify token formats in this profile.

Require Active Browser Sessions

Problem	Should the profile require the STS to only issue tokens if the user has an active browser session with an Identity Provider?!
Assumptions	Tokens are used for invoking services on the user's behalf. In some situations, the WSP may want assurance that the user is currently active (has a session with the IdP) – especially when trust in the WSC is low. Enforcing such a policy would provide the WSP the desired assurance.
Alternatives	1) Require the STS to only issue tokens if the user has a valid session with an Identity Provider. 2) Require the STS only to issue tokens if the user's session was established recently (e.g. based on authentication instant or SessionNotOnOrAfter in the SSO assertion). 3) Leave such policies outside the profile.
Analysis	The real business needs will probably vary among deployments and it can potentially be complicated for an STS to check whether the user's SSO session with the IdP is valid. Another way for the WSP to be assured that the WSC is not abusing a token is to request user interaction (e.g. using the Liberty RedirectRequest protocol) such that the user himself can confirm or provide consent. Further, the IdP session requirement does not make sense in "rich client" scenarios that should also be supported by the profile.
Decision	Leave it as a policy outside the profile.

Binding

Problem	Which bindings should be used for conveying WS-Trust messages?!
Assumptions	We need a widely available binding offering adequate security including support for security tokens.
Alternatives	1) Liberty ID-WSF 2.0 SOAP Binding 2) Liberty Simple SOAP Binding for eGovernment OIO Basic Security Profile 3) OWSA Model T 4) Specify own binding based on WS-Security. 5) Don't specify binding – leave it to deployment profiles.
Analysis	The SWITCH profile uses Liberty SOAP Binding which is the far most commonly deployed among the alternatives. However, some COTS STS products will not support this binding. Thus, it may be the case that STS'es are deployed with different requirements for bindings.
Decision	Don't specify binding – leave it to deployment profiles such as [OIO-WST-DEP].

Token life time

Problem	Should the profile specify token life time and whether the token can be used for multiple service invocations?!
Assumptions	
Alternatives	1) Require the STS to set token life time to a pre-defined value. 2) Restrict tokens for one service invocation. 3) Leave token life time policies outside the profile.
Analysis	From a security perspective, tokens should be fresh and limited in scope to prevent misuse. On the other hand the overhead of obtaining and validating tokens can be considerable – especially in scenarios with multiple web service invocations from one WSC to one WSP on behalf of the same user. Security versus performance trade-offs will probably vary with deployments.
Decision	Leave token life time policies outside the profile. It should be determined in local deployments.

Placement of user token

Problem	Where should the user token be placed when a Web Service Consumer requests a token on behalf of a user.
Assumptions	The STS needs both the identity of the user and Web Service Consumer.
Alternatives	1. Place it in the <wst:OnBehalfOf> element. 2. Use the <wst14:ActAs> defined in WS-Trust 1.4. 3. Define a custom element (non-WST namespace) for the token.
Analysis	After discussions with experts on the <wst:OnBehalfOf> it has become clear that this element does not support the semantics required by this profile: that someone is acting on behalf of a user. Instead <wst:OnBehalfOf> is used when the requestor is a proxy that obtains the token on the Web Service Consumer's behalf and forwards the token back to the WSC for his own use - which is not the case in our eGovernment scenarios. WS-Trust 1.4 has introduced on <wst14:ActAs> element with the semantics we seek – i.e. the WSC requesting a token on the user's behalf. Instead of defining our own custom element for this, we have chosen to adopt the <wst14:ActAs> from the WS-Trust 1.4, which is completely backwards-compatible to WS-Trust 1.3.
Decision	Use the <wst14:ActAs> element from WS-Trust 1.4.

References

[WST] "WS-Trust 1.3", OASIS Standard, 19 March 2007.

[OIO-WST-DEP] "OIO WS-Trust Deployment Profile Version 1.0", Danish IT and Telecom Agency.

[SAML-CORE] "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005.

[OIO-SAML-SSO] "OIO Web SSO Profile V2.0", Danish IT and Telecom Agency.

[LIB-SOAP] "Liberty ID-WSF SOAP Binding Specification", version 2.0, Liberty Alliance Project.

[Scenarios] "Identity-Based Web Services – Scenarios", Danish IT and Telecom Agency.

[SWITCH] "WS-Trust 1.3 Interoperability Profile", Working Draft 01, 07 January 2008.

[ISIP] "Identity Selector Interoperability Profile V1.5", Microsoft Corporation.