>

OIO SAML Profile for Identity Tokens

Version 1.0

IT- & Telestyrelsen
October 2009

Content >

Document History

Date	Version	Initials	Changes
09-06-2009	0.9	SPN	Specified only supported SAML confirmation method to be Holder-of-Key. Document ready for OIO public hearing
21-09-2009	1.0	TG	Document updated after public hearing. Assertions are now allowed to be encrypted entirely using the `<EncryptedAssertion>` element in order to better support scenarios with strong privacy requirements.

Introduction

This document specifies a SAML 2.0 profile for identity tokens to be used in context of identity-based web services. The profile re-uses a number of elements from the OIO SAML profile for Web SSO described in [OIO-SAML-SSO].

A number of scenarios containing extensions to the web SSO scenarios can be found in [Scenarios]. In all these scenarios, a service provider needs to invoke a remote identity-based web service in order to service the user. For this purpose the user identity is passed in the web service call in a SAML assertion called an identity token (specified in this profile). Thus, the *invocation entity* (the user) is different from the *sender* (the web service consumer).

The identity token is different from the authentication assertion established during web SSO in a number of ways:

- The goal is not to establish a browser session but instead to hand the web service provider a user context for the web service invocation.
- The subject confirmation element may bind the assertion to the web service consumer's signature holder-of-key assertions).
- The identity token is not intended for the service provider who has an active session with the user – but for the provider of the identity-based web service (web service provider). This has implications for audience, encryption and name identifiers. For example, the user may have different identifiers (e.g. pseudonyms) at different service providers.
- The token will be requested using a profile of the WS-Trust standard (see [OIO-WST]) and issued by a Security Token Service.

The primary goal of an identity token is thus simply to convey a set of identity attributes about a user.

Related profiles

This profile is designed to be "compatible" with the following profiles:

- The Liberty Alliance ID-WSF 2.0 SecMech SAML Profile [LIB-SAML].
- The OASIS Web Services Security SAML Token Profile 1.1 [WSS-SAML].

Thus, identity tokens conforming to the requirements below should be usable with both these profiles and therefore the WS-* and Liberty stacks in general.

A number of other documents and profiles are closely related:

- The [Scenarios] document describes the overall business goals and requirements and shows how the different OIO profiles are combined to achieve these.
- The OIO WS-Trust Profile shows how to request and retrieve identity tokens from a secure token service (STS) [OIO-WST].
- The OIO Web SSO SAML profile [OIO-SAML-SSO] specifies a SAML 2.0 profile for web SSO which is used to "bootstrap" identity-based web services. The SAML authentication assertions described in there may contain Identity Provider end point references and bootstrap tokens which can be used to retrieve identity tokens described in this profile.

The reader is assumed to be familiar with the existing web SSO profile [OIO-SAML-SSO].

Profile Requirements

>

This profile specifies a number of requirements for SAML Assertions issued by Security Token Services for subsequent use as identity tokens. Notice that neither SAML protocols nor bindings are relevant for the profile since identity tokens will be used as message elements in other protocols having their own bindings.

<Assertion> Requirements

- The `<Issuer>` element MUST contain the unique identifier of the issuing entity (i.e. Security Token Service). The Format attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.
- The Issuer ID MUST contain a Uniform Resource Locator containing the issuer's domain.
- The assertion MUST contain exactly one `<AttributeStatement>` element and one `<AuthnStatement>`.
- The assertion MUST NOT contain an `<AuthzDecisionStatement>.`
- The assertion MUST be signed by the STS by including a `<ds:Signature>` element. The private key used for signing MUST be bound to the STS's X.509 certificate.
- An assertion MUST contain exactly one `<Subject>` representing the identity associated with the token.
- When the identity token is used in a web service call, the `<Subject>` element MUST represent the invocation entity (i.e. the entity on whose behalf the web service call is made).
- The assertion MAY be encrypted using the `<saml2:EncryptedAssertion>` element. This can e.g. be used when the requester (WSC) should not learn the user's identity at the destination service (WSP) – for example when pseudonyms are used.
- The Name identifier element MAY be of type `<saml2:EncryptedID>` when the subject identity is only to be disclosed to the relying party.
- Encryption of the entire assertion or NameID element SHOULD always be applied if pseudonyms are used.
- The subject element MUST contain at least one `<SubjectConfirmation>` element with a confirmation method of
    - `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key`
- When holder-of-key subject confirmation is used:
    - The element MUST be qualified with an `xsi:type` of `saml2:KeyInfoConfirmationDataType`
    - Exactly one `<ds:KeyInfo>` element MUST be included containing a `<ds:X509Data>` and a `<ds:X509Certificate>` with the X.509 certificate of the sender as a base64 encoded value.

  - If sender / attesting entity (e.g. a Web Service Consumer) is different from the Subject of the assertion, this MUST be identified in the `<saml2:SubjectConfirmation>` element using a separate `<saml2:NameID>` element with a `Format` attribute value of `urn:oasis:tc:SAML2:2.0:nameid-format:entity`. The value SHOULD identify the attesting entity to the recipient and MAY contain a SAML entity ID, a service address (e.g. the content of the `<wsa:Address>` in the `<wsa:ReplyTo>` header) or identity from the sender's certificate (e.g. Distinguished Name).

- The `<SubjectConfirmation>` element SHOULD include a `NotOnOrAfter` attribute. After this instant the assertion MUST be considered invalid. Relying parties MAY reject identity tokens based on stricter local policies regarding life time of assertions (time since the assertion's `IssueInstant`).

- The assertion MUST contain an `<AudienceRestriction>` including the intended recipient's unique identifier as an `<Audience>.`

- Advice elements MAY safely be ignored by implementations. This implies that provider chaining information as specified in [LIB-SAML] SHOULD not be included in the element.

<AttributeStatement> Requirements

- The `<AttributeStatement>` element MUST follow the naming and encoding rules for attributes defined in [OIO-SAML-SSO]. It is not required to include all attributes defined in OIOSAML since the requester of the token may specifically have indicated which claims that are needed. For a mapping between claims in the WS-Trust request and attributes in the issued tokens, please see the OIO WS-Trust Deployment Profile [OIO-WST-DEP].
- The `AssuranceLevel` attribute is mandatory MUST reflect how the user authenticated initially to the Identity Provider. Thus, it MUST have the same value as in authentication assertions issued for Web SSO.

- If an attribute values require confidentiality and the assertion is not encrypted at the outer level through a `<saml2:EncryptedAssertion>` element, the attribute element SHOULD be of type `<saml2:EncryptedAttribute>`[1].

- The assertion issuer MAY limit the resource which the invoker may access at the relying party by describing relevant resources in the `<saml2:AttributeStatement>` or by including attributes describing the subject's roles. Such attributes will not be part of this profile and SHOULD be agreed bilaterally between issuer and relying party. The attribute encoding rules defined in [OIO-SAML-SSO] MUST be followed.

[1] Note that this overrides [OIO-SAML-SSO] which requires encryption of the entire assertion.

```
                                                                          >
```

SAML 2 Token Processing

When an assertion conforming to this profile is used within a web service request, the following steps should be taken during validation by the recipient:

- The recipient MUST verify that the message was not issued after the time indicated in the `NotOnOrAfter` conditions (subject to allowed clock skew).
- The recipient MUST verify that it is listed as an intended audience in the `<saml2:AudienceRestriction>` element.
- The signature on the assertion MUST be validated as described in the SAML 2 specification. Requirements for checking the revocation status of certificates including the allowed methods (CRL, OCSP etc.) is left as a policy decision.
- If the assertion employs a holder-of-key confirmation method the token service MUST verify that the requester is in possession of the private key.

Security Requirements

>

Note that requirements for signing and encrypting SAML messages or elements have already been listed in previous sections.

Certificates

- For signing and encryption of messages, X.509 certificates MUST be used. It is left to federations using the profile to determine the allowed types of certificates (and hence trust mechanisms).

Encryption Algorithms

- Encryption algorithm MUST be AES with at least 128 bit keys.
- Signature algorithm MUST be SHA1withRSA or SHA256withRSA with minimum 1024 bit modulus.
- Longer AES or RSA keys are permitted.
- When using 1024 bit RSA modulus, federation participants SHOULD prepare to upgrade a longer modulus within 6-24 months.

Security Considerations

This profile is not known to introduce any new security issues not described in the underlying profiles. We refer to [LIB-SAML], [WSS-SAML] and [SAML-CORE] for details.

Compatibility with other profiles and frameworks

>

As mentioned in the introduction this profile is designed to be compatible with the Liberty ID-WSF 2.0 SecMech SAML Profile [LIB-SAML] and OASIS WSS SAML Token Profile 1.1 [WSS-SAML].

However, there is one issue worth mentioning: the Liberty profile requires identity tokens to include an endpoint reference for the Liberty Discovery Service associated with the subject identity. Since the present identity token profile is designed to be used in scenarios without such discovery services (i.e. an STS will be used instead), this requirement has not been reflected in this profile. See [Scenarios] for an overview of the scenarios.

Whether this difference will create any interoperability issues is unknown and should be investigated in proof-of-concepts. It is believed to be an important issue in combining the WS-* and Liberty web service stacks – for example using identity tokens issued by a Security Token Services in a web service call to a Liberty-enabled web service provider. One solution can be that the token issuer has knowledge on which scheme the recipient prefers and constructs the token accordingly.

Profile and Architectural Decisions

>

This chapter presents the rationale behind important profile decisions. The descriptions are not normative but attempts to provide the reader with some insights to why the profile is designed the way it is.

Encryption of Assertions

Problem	Should identity assertions be encrypted entirely as in the web SSO profile or should attribute-level encryption be used?!
Assumptions	The assertion may contain sensitive data. For example, the user might have different identifiers (pseudonyms) with different service providers and may not want the service provider requesting the identity token to learn the identifier used at the other service provider for which the token is destined.
Alternatives	• Encrypt entire assertion. • Encrypt only parts such as NameID and attributes. • No encryption – assertion is secured using transport security mechanisms (e.g. SSL/TLS or WS-Security message encryption).
Analysis	Transport level encryption will not keep the assertion contents confidential from the requester for the token so this option is not viable. Encrypting the entire assertion under the recipient's public key makes the assertion unreadable by the service provider who requests it. This solves privacy problems but may however introduce new problems. For example, the requesting service provider may not know if there are subject confirmation obligations he should honor in the web service call e.g. proving possession of a key. Another issue with encrypting the entire assertion is that the [LIB-SAML] profile prefers per-attribute encryption and interoperability with the Liberty web service stack could be an issue. Attribute-level encryption solves privacy problems but puts more processing overhead on the recipient (e.g. several decryption operations may be needed).
Decision	Allow entire assertion as well as individual name IDs and attributes to be encrypted.

Subject Confirmation

Problem	Which subject confirmation method should be allowed?!
Assumptions	
Alternatives	• Bearer • Holder of key

	• Sender vouches
Analysis	Bearer assertions have the advantage of being simple to implement (e.g. few processing rules) and may be well-suited for basic scenarios. However, there is a potential that the assertion can be misused by a third party as it is not bound to the sender or message. Sender-vouches assertions do not seem to offer any advantages over bearer assertions in our scenarios since signing is always used. Holder-of-key assertions allow the assertion to be bound to a particular service provider which reduces the risk of misuse. Furthermore, they can be used to establish message authentication and trust since a third party asserts the relation between a key and an identity. This is useful in when trust domains are crossed and the recipient e.g. does not trust the sender's certificate.
Decision	Allow only holder-of-key subject confirmations.

Assertion usage semantics

Problem	Should identity tokens have a one-time usage semantics defined (as Web SSO tokens have)?!
Assumptions	Identity tokens are expensive to retrieve since they require a protocol exchange with a third party.
Alternatives	• Require one-time usage in the profile (via recipient processing rules). • Allow identity tokens to be used several times subject to life-time restrictions (`NotOnOrAfter` attribute).
Analysis	Ideally, a web service consumer should retrieve an identity token for each web service invocation. This would allow fine-grained authorization decisions to take place at the token issuer. Further it will reduce the risk that a token is used after the user has logged out of his browser session with the Identity Provider[2]. On the other hand, retrieving identity tokens for every web service invocation can severely impact performance of the applications.
Decision	Allow the token to be used several times subject to life-time restrictions. The token life time is left as a policy decision such that sensible trade-offs between security/control and performance can be made. Further, the relying party receiving the token is allowed to enforce a stricter time-out policy based on the `IssueInstant` attribute in the assertion. Thus, he may reject tokens that are not yet

[2] We assume here that the Identity Provider and token issuer communicate such that identity tokens can only be issued when the user has an active browser session with the Identity Provider.

	expired.

Include Authentication Statements

Problem	Should identity tokens be allowed to include `<AuthnStatement>` elements?!
Assumptions	The relying party may need to know details of the user authentication at the Identity Provider in order to enforce local authorization policy.
Alternatives	• Allow them. • Do not allow them.
Analysis	The benefit of allowing an `<AuthnStatement>` in identity tokens is that the relying party can see details of the authentication event including the time of authentication and possibly specifics of the authentication mechanism. For example, Liberty [LIB-SAML] allows such statements in identity tokens. On the other hand, such statements introduce additional complexity. In the Danish scenarios, the identity token will always include the `AssuranceLevel` attribute in the `<AttributeStatement>` which is probably what a relying party would want to inspect. In effect, the assurance level is carried over from the authentication token to the identity token.
Decision	Allow `<AuthnStatement>` elements.

Allow authorization information

Problem	Should the token issuer be allowed to include authorization information in the identity token?!
Assumptions	The assertion issuing authority may want to limit the resources which the invoker may access at the relying party by describing relevant resources as part of the token. Alternatively, an assertion issuing authority may want to include attributes describing the subject's roles which can then be used in authorization decisions by the relying party.
Alternatives	• Allow such information to be included as attributes. • Allow such information to be included in an `<AuthzDecisionStatement>` element. • Disallow authorization information.
Analysis	Authorizations or roles may in some scenarios be managed centrally by the component issuing tokens. Including such information in identity tokens may therefore be a handy way to distribute such information. `<AuthzDecisionStatement>` elements are deprecated in SAML 2.0 and it is therefore unwise to use them. Including authorization information in attributes is better and will probably not break any

	implementations that do not understand the attributes. Since authorizations and roles may vary with context the format of such attributes will have to be agreed outside this profile.
Decision	Allow authorization information to be included only as attributes.

Include information on provider chaining?!

Problem	Should the token issuer be allowed to include information on provider chaining in the identity token?!
Assumptions	
Alternatives	• Allow • Disallow
Analysis	The Liberty SAML profile [LIB-SAML] describes how provider chaining information can be included via the `<saml2:Advice>` element: “Provider chaining refers to scenarios in which a service provider (WSP), upon receiving a request from a sender, itself passes the request onto another service provider until the destination service provider is reached... When more than two web service providers are in the chain, information about the earlier web service providers may need to be explicitly recorded to enable the destination web service to make an appropriate authorization decision.” The element introduces added complexity and the Danish scenarios [Scenarios] do not seem to need it. While it is definitely possible that web services can chain we do not foresee advanced authorization policies taking the provider chain into account. Furthermore, provider chaining attributes are Liberty-specific and will not be understood by non-Liberty implementations. Thus, if just one service provider in the chain (or the token issuer) is not Liberty-compliant, this would not work. Thus, in order to achieve interoperability with the WS-* stack it is better to disallow this element.
Decision	Do not allow information on provider chaining to be included in the token.

References

[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.

[OIO-BSP] “OIO Basic Security Profile V1.2”, Danish IT and Telecom Agency.

[OIO-WST] “OIO WS-Trust Profile V1.0”, Danish IT and Telecom Agency.

[OIO-WSP-DEP] “OIO WS-Trust Deployment Profile V1.0”, Danish IT and Telecom Agency.

[LIB-SAML] “ID-WSF 2.0 SecMech SAML Profile”, Liberty Alliance.

[WSS-SAML] “Web Services Security: SAML Token Profile 1.1”, OASIS Standard, 1 February 2006.

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