Liberty ID-WSF Security Mechanisms Core

Version: v2.0-12

Editors:
Gary Ellison, Sun Microsystems, Inc.
Frederick Hirsch, Nokia Corporation
Paul Madsen, Entrust, Inc.

Contributors:
Robert Aarts, Trustgenix
Carolina Canales-Valenzuela, Ericsson
Conor Cahill, America Online, Inc.
Scott Cantor, Internet2, The Ohio State University
Darryl Champagne, IEEE-ISTO
Jeff Hodges, Neustar
John Kemp, Nokia Corporation
John Linn, RSA Security Inc.
Jonathan Sergent, Sun Microsystems, Inc.
Greg Whitehead, Trustgenix, Inc.

Abstract:

Specification from the Liberty Alliance Project Identity Web Services Framework for describing security mechanisms for authentication and authorization.

Filename: draft-liberty-idwsf-security-mechanisms-core-v2.0-12.pdf
Notice

This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the
document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works
of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact
the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this document may require licenses under third party intellectual property
rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are
not, and shall not be held responsible in any manner for identifying or failing to identify any or all such third party
intellectual property rights. This Specification is provided "AS IS", and no participant in the Liberty Alliance
makes any warranty of any kind, express or implied, including any implied warranties of merchantability,
non-infringement of third party intellectual property rights, and fitness for a particular purpose. Implementors
of this Specification are advised to review the Liberty Alliance Project’s website (http://www.projectliberty.org/) for
information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance
Management Board.

Copyright © 2005 Adobe Systems; America Online, Inc.; American Express Company; Amsoft Systems Pvt Ltd.;
Avatier Corporation; Axalto; Bank of America Corporation; BIPAC; BMC Software, Inc.; Computer Associates
International, Inc.; DataPower Technology, Inc.; Diversinet Corp.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.;
Ericsson; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le
développement de l’administration électronique (ADAE); Gamefederation; Gemplus; General Motors; Giesecke &
Devrient GmbH; GSA Office of Governmentwide Policy; Hewlett-Packard Company; IBM Corporation; Intel
Corporation; Intuit Inc.; Kantega; Kayak Interactive; MasterCard International; Mobile Telephone Networks (Pty)
Ltd; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nippon Telegraph and Telephone Corporation; Nokia
Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation;
Reactivity Inc.; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp Laboratories of America;
Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.;
Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Trusted Network Technologies; Trustgenix; UTI; VeriSign, Inc.;
Vodafone Group Plc.; Wave Systems Corp. All rights reserved.

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08855-1331, USA
info@projectliberty.org
## Contents

1. Introduction ................................................................. 4
2. Overview of Identity-Based Web Services Authentication and Authorization (Informative) ............... 5
3. Notation and Terminology .................................................. 7
   3.1. Notational Conventions ............................................... 7
   3.2. Namespace ............................................................. 7
   3.3. Terminology ......................................................... 10
4. Security Requirements (Informative) ........................................ 11
   4.1. Security Requirements Overview ..................................... 11
   4.2. Common Requirements ............................................... 11
   4.3. Peer Authentication Requirements ................................... 12
   4.4. Message Correlation Requirements ................................... 12
   4.5. Privacy Requirements ................................................ 12
   4.6. Service Availability Requirements .................................... 12
5. Confidentiality and Privacy Mechanisms ..................................... 14
   5.1. Transport Layer Channel Protection .................................... 14
   5.2. Message Confidentiality Protection .................................... 14
   5.3. Identifier Privacy Protection .......................................... 14
   5.3.1. Encrypted Name Identifiers ....................................... 15
6. Authentication Mechanisms .................................................. 16
   6.1. Authentication Mechanism Overview (Informative) ..................... 18
   6.2. Peer Entity Authentication ............................................ 19
      6.2.1. Unilateral Peer Entity Authentication ........................... 19
      6.2.2. Mutual Peer Entity Authentication ............................... 20
   6.3. Message Authentication ............................................... 20
      6.3.1. Token Container .................................................. 21
      6.3.2. Common Sender Processing Rules ................................. 22
      6.3.3. Common Recipient Processing Rules ............................... 23
   6.4. WSS X.509 Token Authentication ....................................... 24
      6.4.1. Sender Processing Rules ......................................... 24
      6.4.2. Recipient Processing Rules ...................................... 25
      6.4.3. X.509 v3 Message Authentication .................................. 25
   6.5. Bearer Token Authentication ........................................... 26
      6.5.1. Sender Processing Rules ......................................... 27
      6.5.2. Recipient Processing Rules ...................................... 27
      6.5.3. Binary Security Token Bearer Tokens ............................. 28
   6.6. Identity Tokens ....................................................... 29
7. Message Authorization Mechanisms ........................................ 31
   7.1. Authorization Mechanism Overview (Informative) ..................... 31
   7.2. Authorization Assertion Generation .................................... 31
   7.3. Provider Chaining .................................................... 31
      7.3.1. Supporting Schema .............................................. 33
   7.4. Presenting Authorization Data ........................................ 35
      7.4.1. Processing Rules ............................................... 35
   7.5. Consuming Authorization Data ......................................... 35
      7.5.1. Processing Rules ............................................... 35
8. Schema ............................................................................. 36
9. References ........................................................................... 38
1. Introduction

This document specifies security protocol mechanisms for securing the consumption of identity-based web services. An identity-based web service is a particular type of a web service that acts upon some resource to either retrieve information about an identity, update information about an identity, or to perform some action for the benefit of some identity. This document describes authentication mechanisms which are factored into the authorization decisions enforced by a given identity-based web service. The specified mechanisms provide for authentication, signing and encryption operations. XML-Signature [XMLSignature] and XML-Encryption [XMLEncryption] are utilized to provide the associated transformations and processing semantics to accommodate the message authentication and protection functionality. OASIS Web Services Security SOAP Message Security [OASISWSSM] compliant header elements communicate the relevant security information, i.e., a SAML [SAMLCore1] or [SAMLCore2] assertions, along with the protected message. Separate SechMech profiles are defined for the use of specific security tokens in conjunction with this core document [LibertySecMech2SAML].
2. Overview of Identity-Based Web Services Authentication and Authorization (Informative)

This document describes security mechanisms that may be used in conjunction with identity-based web services defined by the Liberty Alliance standards. An identity-based web service is a particular type of a web service that acts upon some resource to retrieve information about an identity, update information related to an identity, or perform some action for the benefit of some identity. A resource is either data related to some identity or a service acting for the benefit of some identity. Although this specification focuses on identity-based services, this does not imply that these mechanisms may not also be used with other web services or that identity and non-identity based web service requests may not be combined as needed by applications.

This specification assumes a model with the following parties: an invoker, a requester, a discovery service and a service provider. An invoker is a principal whose identity is related to an identity-based service. A requester is a web services client that is making a service request. In many cases the requester is the same as the invoker, as in the case where a web service client makes a web service request related to its own identity. An example where the invoker is distinct from the requester is when a browser based client invokes an identity-based web service by delegating the request to a web service client. In this case this requester acts on behalf of the browser client. The service provider offers an identity-based web service and responses to web service requests. The Discovery Service provides a service endpoint reference and possibly security tokens to the requester to enable the requester to reach the service provider that offers the identity-based service.

In many cases, the requester directly interacts with the identity-based web service, and the identity-based web service implements both the authorization policy decision point (PDP) and policy enforcement point (PEP). Under these circumstances the authorization decision should be made according to the policies of the service provider and MAY be based on the identity of the invoker, the identity of the requester, the authentication context of the requester, the specific resource being accessed, and other information known to the provider. In order to make a request to the service provider, the requester may obtain a service endpoint reference from a Discovery Service. In this case the Discovery Service may also make an authorization decision, and refuse to provide a service endpoint reference for services that are not authorized by the Discovery Service.

In the case of delegation, the invoker may provide the requester with credentials that may be used in authorization decisions. In this case an authentication assertion for the invoker may be included in the service request, allowing the authorization decision at the service provider to be based not only on the identity of the service requester (the portal), but also the invoker (the browser client). Such an assertion may be obtained through a SAML 2.0 profile that enables authentication of the browser client to the service requester, or using a single sign-on service as outlined in the Liberty ID-WSF Authentication Service and Single Sign-On Specification.

To access an appropriate identity-based service, a web service requester must first obtain a service endpoint reference from a discovery service for the appropriate service provider. Which is appropriate is determined by the discovery service, which knows which services are available, and it authorizes the service requester to contact. The service endpoint reference may include the following:

- A list of allowed authentication mechanisms for interacting with the service provider. The service endpoint reference includes a list of authentication mechanism identifiers that each specify an allowed combination of peer and message level authentication. These identifiers are defined in this specification.

- Security token instances that the client may use to access the service provider. Such tokens may include authentication or authorization tokens provided by the discovery service.

- Additional information relevant to future authorization decisions, such as the path through proxies taken by the request so far. The discovery service may include such information in a security token, as described in this specification.
This specification also defines identity tokens, tokens that are used to convey additional identity information for a party that is part of a transaction, but not necessarily the invoker and may not be present. The service provider may need to make authorization decisions based on this additional information. An example is when Bob accesses a photo service to access Alice’s photos - Alice may not be present but her identity may need to be presented by Bob using an identity token.

To summarize, access to an identity-based web service may be controlled at one or more points. One point is the discovery service, which will only provide service endpoint references that are appropriate to the invoker and requester. Another is at the service provider itself, which may also perform authorization decisions based on its knowledge and the tokens presented to it with a request.

Material specific to specific tokens is in the Security Mechanism token profiles, in particular the SAML token profile [LibertySecMech20SAML].
3. Notation and Terminology

This section specifies the notations, namespaces and terminology used throughout this specification. This specification uses schema documents conforming to W3C XML Schema (see [Schema1]) and normative text to describe the syntax and semantics of XML-encoded messages.

3.1. Notational Conventions

Note: Phrases and numbers in brackets [ ] refer to other documents; details of these references can be found in the References.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

3.2. Namespace

The following namespaces are referred to in this document:
Table 1. Namespaces
<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This namespace is used for Liberty Security Mechanisms in ID-WSF 2.0 and is defined in this document and earlier interim releases of this document.</td>
</tr>
<tr>
<td>sb:</td>
<td>urn:liberty:sb:2005-11</td>
</tr>
<tr>
<td></td>
<td>This namespace represents the Liberty SOAP Binding namespace (v2.0). It is defined in the Liberty SOAP Binding document, v2.0 [LibertySOAPBinding].</td>
</tr>
<tr>
<td></td>
<td>This namespace represents the Liberty discovery service. It is defined in [LibertyDisco].</td>
</tr>
<tr>
<td>saml:</td>
<td>urn:oasis:names:tc:SAML:1.0:assertion</td>
</tr>
<tr>
<td></td>
<td>This namespace represents SAML 1.0 assertions. It is defined in [SAMLCore11].</td>
</tr>
<tr>
<td>saml2:</td>
<td>urn:oasis:names:tc:SAML:2.0:assertion</td>
</tr>
<tr>
<td></td>
<td>This namespace represents SAML 2.0 assertions. It is defined in [SAMLCore2].</td>
</tr>
<tr>
<td>S:</td>
<td><a href="http://www.w3.org/2002/12/soap-envelope">http://www.w3.org/2002/12/soap-envelope</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the SOAP 1.2 namespace. It is defined in [SOAPv1.2].</td>
</tr>
<tr>
<td>ds:</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#">http://www.w3.org/2000/09/xmldsig#</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the XML Signature namespace. It is defined in [XMLDsig].</td>
</tr>
<tr>
<td>xenc:</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#">http://www.w3.org/2001/04/xmlenc#</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the XML Encryption namespace. It is defined in [xmlenc-core].</td>
</tr>
<tr>
<td>wsa:</td>
<td><a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the WS-Addressing namespace. It is defined in [WSAv1.0].</td>
</tr>
<tr>
<td>wsse:</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the SOAP Message Security namespace. It is defined in [wss-sms].</td>
</tr>
<tr>
<td>wsu:</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the SOAP Message Security Utility namespace. It is defined in [wss-sms].</td>
</tr>
<tr>
<td>xs:</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
</tr>
<tr>
<td></td>
<td>This namespace represents the W3C XML schema namespace. It is defined in [Schema1].</td>
</tr>
</tbody>
</table>
This specification uses the following typographical conventions in text:

- Elements and attributes: `<Element>`
- Data types: A `datatype`
- Constants: A `constant`
- Code: `<saml2:AuthnStatement...>`

For readability, when an XML Schema type is specified to be xs:boolean, this document discusses the values as true and false rather than "1" and "0".

### 3.3. Terminology

Definitions for Liberty-specific terms can be found in [LibertyGlossary].

The following terms are defined below as an aid in understanding the participants in the message exchanges:

- **Recipient** – entity which receives a message that is the ultimate processor of the message
- **Sender** – the initial SOAP sender. A sender is a proxy when its identity differs from the invocation identity.
- **Proxy** – entity whose authenticated identity, according to the recipient, differs from that of the entity making the invocation.
- **Trusted Authority** – a Trusted Third Party (TTP) that issues, and vouches for, SAML assertions
- **Invocation Identity** – party invoking a service.
- **Service** – invocation responder, providing a service. Ultimate message processor.
4. Security Requirements (Informative)

This section details the security requirements that this specification must support. This section first presents use case scenarios envisioned for identity-based web services. We then follow-up the discussion with the requirements the usage scenarios prescribe.

4.1. Security Requirements Overview

There are multiple facets this security specification considers:

- Authentication of the sender
- When the sender is not the invocation identity, the proxy rights for sender to make a request on behalf of invocation identity
- Authentication of the response
- Authentication context and session status of the interacting entity
- Authorization of invocation identity to access service or resource

Note that the authorization mechanism draws a distinction between the invocation identity and the identity of the initial SOAP sender making a request to the identity web service. These two identities are referred to as the invocation identity and the sender identity, respectively. In effect, this enables a constrained proxy authorization model.

The importance of the distinction between invocation and sender identity lies in the service’s access control policies whereby the service’s decision to grant or deny access may be based on either or both identities. The degenerate case is where the invocation identity is the same as the sender identity, in which case no distinction need be made.

Note that a browser-based user agent interacting with some service provider does not necessarily imply that the service provider will use the user identity as the invocation identity. In some cases, the identity of the service provider may still be used for invocation.

The above scenarios suggest a number of requirements in order to secure the exchange of information between participants of the protocol. The following list summarizes the security requirements:

- Request Authentication
- Response Authentication
- Request/Response Correlation
- Replay Protection
- Integrity Protection
- Confidentiality Protection
- Privacy Protections
- Resource Access Authorization
- Proxy Authorization
- Mitigation of denial of service attack risks
4.2. Common Requirements

The following apply to all mechanisms in this specification, unless specifically noted by the individual mechanism.

- Messages may need to be kept confidential and inhibit unauthorized disclosure, either when in transit or when stored persistently. Confidentiality may apply to the entire message, selected headers, payload, or XML portions depending on application requirements.

- Messages may need to arrive at the intended recipient with data integrity. SOAP intermediaries may be authorized to make changes, but no unauthorized changes should be possible without detection. Integrity requirements may apply to the entire message, selected headers, payload, or XML portions depending on application requirements.

- The authentication of a message sender and/or initial sender may be required by a receiver to process the message. Likewise, a sender may require authentication of the response.

- Protection against replay or substitution attacks on requests and/or responses may be needed.

- The privacy requirements of the participants with respect to how their information is shared or correlated must be ensured.

4.3. Peer Authentication Requirements

The security mechanisms supported by this framework must allow for active and passive intermediaries to participate in the message exchange between end entities. In some circumstances it is necessary to authenticate all active participants in a message exchange.

Under certain conditions, two separate identities must be authenticated for a given request: the invocation identity and the sender identity. The degenerate case is where the identity of the message sender is to be treated as the invocation identity, and thus, no distinction between invocation identity and sender identity is required. In support of this scenario the candidate mechanism to convey identity information is client-side X.509 v3 certificates based authentication over a SSL 3.0 (see SSL) or TLS 1.0 (see RFC2246) connection. Generally, this protocol framework may rely upon the authentication mechanism of the underlying transfer or transport protocol binding to convey the identity of the communicating peers.

However for scenarios where the sender’s messages are passing through one or more intermediaries, the sender must explicitly convey its identity to the recipient by using a WSSEc token profile which specifies processing semantics in support of Proof-of-Possession. For example, the Web Services Security SAML Token Profile defines Proof-of-Possession processing semantics [wss-saml11]. Other possible bindings include Kerberos whereby the session key is used to sign the request.

4.4. Message Correlation Requirements

The messages exchanged between participants of the protocol MAY require assurance that a response correlates to its request. This may require integrity protection.

4.5. Privacy Requirements

Adequate privacy protections must be assured so as to inhibit the unauthorized disclosure of personally identifiable information. In addition, controls must be established so that personally identifiable information is not shared without user notification and consent and that where applicable privacy regulations may be accommodated. This may require prescriptive steps to prevent collusion among participants in an identity network.

4.6. Service Availability Requirements
The system must maintain availability, requiring the implementation of techniques to prevent or reduce the risk of attacks to deny or degrade service.

4.7. Resource Access Authorization Requirements

Previously we mentioned the notion of conveying both a *sender identity* and an *invocation identity*. In doing so the framework accommodates a restricted proxy capability whereby a provider of an identity-based web service (the intermediate system entity or proxy) can act on behalf of another system entity (the subject) to access an identity-based web service (the recipient). To be granted the right to proxy for a subject, the intermediate system entity may need to interact with a trusted authority. Based on the authority’s access control policies, the authority may generate and return an assertion authorizing the provider to act on behalf of the subject to the recipient. This protocol framework can only convey authoritative information regarding the identities communicated to other system entities. Even with the involvement of a trusted authority that makes authorization decisions permitting a provider to access a web service on behalf of another party, the final service provider should still implement a policy enforcement point.
5. Confidentiality and Privacy Mechanisms

Some of the service interactions described in this specification include the conveyance of information that is only known by a trusted authority and the eventual recipient of a resource access request. This section specifies the schema and measures to be employed to attain the necessary confidentiality controls.

5.1. Transport Layer Channel Protection

When communicating peers interact directly (i.e. no active intermediaries in the message path) then transport layer protection mechanisms may suffice to ensure the integrity and confidentiality of the message exchange.

• Messages between sender and recipient MUST have their integrity protected and confidentiality MUST be ensured. This requirement MUST be met with suitable SSL/TLS cipher suites. The security of the SSL or TLS session depends on the chosen cipher suite. An entity that terminates an SSL or TLS connection needs to offer (or accept) suitable cipher suites during the handshake. The following list of TLS 1.0 cipher suites (or their SSL 3.0 equivalent) is RECOMMENDED.

  • TLS_RSA_WITH_RC4_128_SHA
  • TLS_RSA_WITH_3DES_EDE_CBC_SHA
  • TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

The above list is not exhaustive. The recommended cipher suites are among the most commonly used. New cipher suites using the Advanced Encryption Standard have been standardized by the IETF [RFC3268] and are just beginning to appear in TLS implementations. It is anticipated that these AES-based cipher suites will be widely adopted and deployed.

  • TLS_RSA_WITH_AES_CBC_SHA
  • TLS_DHE_DSS_WITH_AES_CBC_SHA

For signing and verification of protocol messages, communicating entities SHOULD use certificates and private keys that are distinct from the certificates and private keys applied for SSL or TLS channel protection.

• Other security protocols (e.g. Kerberos, IPSEC) MAY be used as long as they implement equivalent security measures.
5.2. Message Confidentiality Protection

In the presence of intermediaries, communicating peers MUST ensure that sensitive information is not disclosed to unauthorized entities. To fulfill this requirement, peers MUST use the confidentiality mechanisms specified in [wss-sms] to encrypt the SOAP envelope <S:Body> content.

Please note that this mechanism does not fully address the privacy and confidentiality requirements of information supplied by a trusted authority which is subsequently carried in the <S:Header> which is not to be revealed to the entity interacting with the recipient. For example the authorization data may contain sensitive information. To accommodate this requirement the trusted authority and ultimate recipient MUST rely upon the mechanisms specified in Encrypted Name Identifiers (Section 5.3.1).

5.3. Identifier Privacy Protection

Under certain usage scenarios the information conveyed by the Trusted Authority for consumption by the identity-based web service may contain privacy sensitive data. However, this data generally passes through the system entity accessing the particular identity-based web service. One example is the name identifier from the federated namespace of the authority and the identity-based web service. Another sensitive data item may be the target identity header, which may have message level encryption applied for confidentiality (SOAP Message Security encryption).

5.3.1. Encrypted Name Identifiers

The identifier conveyed in the subject MUST be resolvable in the namespace of the consuming service instance. However, this requirement is in conflict with the need to protect the privacy of the identifier when the message passes through intermediaries.

The SecMech profiles describe how to accomplish this.
6. Authentication Mechanisms

This specification defines a set of authentication mechanisms, labeled by URIs, to support various security requirements. Multiple mechanisms are specified to accommodate various deployment scenarios. Authentication may be performed at different protocol layers, or in combination, resulting in different properties. In addition, different mechanisms may be used at each layer. The two authentication layers that are specified in this document include:

- Peer Entity Authentication
- Message Authentication

In each case the URN is constructed in a manner to summarize various information about the mechanism, similar in concept to SSL/TLS CipherSuites. In particular, the URN is created as follows: urn:liberty:security:DATE:PEER:MESSAGE. The DATE is associated with one or more versions of ID-WSF, and is defined in the form yyyy-mm. PEER indicates the kind of peer authentication in effect (if any), and MESSAGE indicates the form of message authentication (if any).

For either of the PEER or MESSAGE properties a value of "null" indicates that the particular security property is not required by the mechanism.

The following DATE values have been defined:

<table>
<thead>
<tr>
<th>DATE</th>
<th>ID-WSF version</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-08</td>
<td>ID-WSF 1.0</td>
</tr>
<tr>
<td>2004-04</td>
<td>ID-WSF 1.0 Errata</td>
</tr>
<tr>
<td>2005-02</td>
<td>ID-WSF 1.1</td>
</tr>
<tr>
<td>2004-12</td>
<td>ID-WSF 2.0 Release 1.0</td>
</tr>
</tbody>
</table>

New version URNs are only defined if necessary, otherwise earlier URNs should be used. Thus for given functionality, the latest version URN should be used appropriate for the ID-WSF release.
The following PEER mechanisms have been defined:

<table>
<thead>
<tr>
<th>PEER</th>
<th>Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>None</td>
</tr>
<tr>
<td>TLS</td>
<td>Peer recipient (SSL/TLS server) authentication</td>
</tr>
<tr>
<td>ClientTLS</td>
<td>Mutual Peer authentication</td>
</tr>
</tbody>
</table>

For the peer entity authentication property, the qualifier indirectly indicates which actor(s) is authenticated in a given interaction.

The following MESSAGE mechanisms have been defined:

<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>None</td>
</tr>
<tr>
<td>SAML</td>
<td>Use of SAML 1.x assertions in conjunction with SOAP Message Security, as outlined in the SecMech SAML 1.x profile.</td>
</tr>
<tr>
<td>SAMLV2</td>
<td>Use of SAML 2.0 assertions in conjunction with SOAP Message Security, as outlined in the SecMech SAML 2 profile.</td>
</tr>
<tr>
<td>X509</td>
<td>SOAP Message Security X509 Token Profile invoker authentication</td>
</tr>
<tr>
<td>Bearer</td>
<td>Bearer token invoker authentication</td>
</tr>
</tbody>
</table>

The MESSAGE authentication qualifier describes the security profile utilized to secure the message. Note that not all message layer authentication mechanisms require the token to be cryptographically bound to the message at the message layer. Bearer tokens, specifically, do not require the token to be bound to the message.

The following table summarizes all the authentication mechanism identifiers defined as of the publication of this specification. Specifically, [SAMLCore11] based identifiers were defined in previous versions of this specification [LibertySecMech11] and [LibertySecMech12].
Table 5. Authentication Mechanisms

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Peer Entity</th>
<th>Message Entity</th>
<th>1.0</th>
<th>1.0 Errata</th>
<th>1.1</th>
<th>2.0 R1</th>
</tr>
</thead>
<tbody>
<tr>
<td>null:null</td>
<td>No</td>
<td>No</td>
<td>2003-08</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>null:X509</td>
<td>No</td>
<td>Yes</td>
<td>2003-08</td>
<td>-</td>
<td>2005-02</td>
<td>-</td>
</tr>
<tr>
<td>null:SAML</td>
<td>No</td>
<td>Yes</td>
<td>2003-08</td>
<td>-</td>
<td>2005-02</td>
<td>-</td>
</tr>
<tr>
<td>null:SAMLV2</td>
<td>No</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2004-12</td>
</tr>
<tr>
<td>null:Bearer</td>
<td>No</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2004-04 2005-02</td>
</tr>
<tr>
<td>TLS:NULL</td>
<td>Recipient</td>
<td>No</td>
<td>2003-08</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TLS:X509</td>
<td>Recipient</td>
<td>Yes</td>
<td>2003-08</td>
<td>-</td>
<td>2005-02</td>
<td>-</td>
</tr>
<tr>
<td>TLS:SAML</td>
<td>Recipient</td>
<td>Yes</td>
<td>2003-08</td>
<td>-</td>
<td>2005-02</td>
<td>-</td>
</tr>
<tr>
<td>TLS:SAMLV2</td>
<td>Recipient</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2004-12</td>
</tr>
<tr>
<td>TLS:Bearer</td>
<td>Recipient</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2004-04 2005-02</td>
</tr>
<tr>
<td>ClientTLS:NULL</td>
<td>Mutual</td>
<td>No</td>
<td>2003-08</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ClientTLS:X509</td>
<td>Mutual</td>
<td>Yes</td>
<td>2003-08</td>
<td>-</td>
<td>2005-02</td>
<td>-</td>
</tr>
<tr>
<td>ClientTLS:SAML</td>
<td>Mutual</td>
<td>Yes</td>
<td>2003-08</td>
<td>-</td>
<td>2005-02</td>
<td>-</td>
</tr>
<tr>
<td>ClientTLS:SAMLV2</td>
<td>Mutual</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2004-12</td>
</tr>
<tr>
<td>ClientTLS:Bearer</td>
<td>Mutual</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2004-04 2005-02</td>
</tr>
</tbody>
</table>

344 1 The bearer token is not bound to the message and is not protected by the TLS mechanism in this case.
345 2 The bearer token is not bound to the message at the SOAP Message layer. It is bound to the message and protected by TLS for a single TLS link, but not protected end-end if the SOAP message traverses SOAP intermediaries.

6.1. Authentication Mechanism Overview (Informative)

The above table depicts the various authentication mechanism identifiers and the authentication properties they exhibit. A description of the setting in which a particular mechanism should be deployed is out of scope for this specification. However, this section describes the characteristics of the class of mechanism and general circumstances whereby the deployment of a given mechanism may be appropriate.

The identifier, urn:liberty:security:2003-08:null:null, does not exhibit any security properties and is defined here for completeness. However one can envision a deployment setting in which access to a resource does not require rigor in authenticating the entities involved in an interaction. For example, this might apply to a weather reporting service.

The peer entity authentication mechanisms defined by this specification leverage the authentication features supplied by SSL 3.0 [SSL] or TLS 1.0 [RFC2246]. The mechanism identifier describes whether the recipient ("TLS") is unilaterally authenticated or whether each communicating peer ("ClientTLS") is mutually authenticated to the other peer. The peer entity authentication mechanisms (Section 6.2) are best suited for direct message exchanges between end systems and when the message exchange may be sufficiently trusted to not require additional attestation of the message payload. However this does not obviate the processing of subject confirmation obligations but rather enables alternative and potentially optimized processing rules. Such optimizations are a matter of security policy as it applies to the trust model in place between communicating entities.

The message authentication mechanisms indicate which attestation profile is utilized to ensure the authenticity of a message. These message authentication facilities aid the deployer in the presence of intermediaries. The different message authentication mechanisms are suited (but not necessarily restricted) to different authorization models.
• The X.509 v3 Certificate mechanism (Section 6.4) is suited for message exchanges that generally rely upon message authentication as the principle factor in allowing the recipient to make authorization decisions.

• The SAML Assertion mechanism (See the SechMech SAML profiles) is suited for message exchanges that generally rely upon message authentication as well as the conveyance and attestation of authorization information in order to allow the recipient to make authorization decisions.

• The Bearer mechanism (Section 6.5) is used to convey the authenticated identity of an invoker with a message. The bearer token need not be bound to the message.

Each operational setting has its own security and trust requirements and in some settings the issuance of bearer tokens by a security token service, such as [LibertyDisco] may greatly simplify the sender’s processing obligations. For example, when the Discovery service indicates that a bearer mechanism is supported and issues a bearer token, the sender can simply populate the security header with the tokens and send the request. However this does not necessarily obviate the requirement for the recipient to process and verify the bearer token. Such an optimization is a matter of security policy as it applies to the trust model in place between the communicating entities.

Not all peer entity authentication and message authentication combinations make sense in a given setting. Again this is a matter of security policy and the trust model the policy accords. For example, in a conventional setting where peer entity authentication is relied upon to ensure the authenticity, confidentiality and integrity of the transport in conjunction with message authentication to assure message authorship, intent and retention of the act of attestation then the mechanism urn:liberty:security:2005-02:ClientTLS:X509 is relevant. However, such a combination may make little sense when peer entity authentication is relied upon to imply message authentication. For example, the mechanism urn:liberty:security:2005-02:ClientTLS:X509 seems equivalent to urn:liberty:security:2003-08:ClientTLS:null in such a setting. A similar argument can be made for the SAML mechanisms (urn:liberty:security:2005-02:ClientTLS:SAML or urn:liberty:security:2004-12:ClientTLS:SAMLv2). The relationship between the identity authenticated as a result of peer entity authentication and the identity authenticated (or implied) from message authentication may diverge and describe two distinct system entities for example, a system principal and a user principal respectively. The identities may also be required to reflect the same system entities. This is a matter of deployment and operational policy and is out of scope for this specification.

6.2. Peer Entity Authentication

The Peer entity authentication mechanisms supported by this specification all rely upon the inherent security properties of the TLS/SSL protocol (sometimes referred to as transport-level security); the different mechanisms are differentiated by how the peers are authenticated. The mechanisms described below have distinct security properties regarding which peers in a message exchange are authenticated. Note that peer authentication may not provide adequate security when SOAP intermediaries are part of the message path and end-to-end security is required. In this case Message level security may be used in place of, or in conjunction with peer entity authentication, as appropriate.

For the mechanisms that include both peer entity authentication and message authentication, optimizations regarding attestation MAY be employed. For example, in environments where there is no requirement that a signature attesting to the authenticity of the message be retained, then it may be sufficient to rely upon the security properties of peer entity authentication to assure the integrity and authenticity of the message payload with no additional message layer signature.

6.2.1. Unilateral Peer Entity Authentication

The semantics and processing rules for mechanisms with PEER having the value of TLS are described in this section. These URIs support unilateral (recipient) peer entity authentication and are of the form: urn:liberty:security:2003-08:TLS:MESSAGE where MESSAGE may vary depending on the message authentication mechanism deployed (e.g. may be null, X509 etc).

The primary function of the TLS mechanism is to provide for the authentication of the receiving entity and to leverage confidentiality and integrity features at the transport layer.
6.2.1.1. Processing Rules

These mechanisms MUST implement TLS/SSL end entity authentication in accordance with the TLS/SSL specifications and employing a cipher suite based on X.509 certificates, requiring the following:

- The sender MUST authenticate the recipient.
- The recipient MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.
- Statements about CipherSuites are provided in section 5.1, Transport Layer Channel Protection.

6.2.2. Mutual Peer Entity Authentication

The semantics and processing rules for mechanisms with PEER having the value of ClientTLS are described in this section. These URIs support mutual (sender and recipient) peer entity authentication and are of the form: urn:liberty:security:2003-08:ClientTLS:MESSAGE where MESSAGE may vary depending on the message authentication mechanism deployed (e.g. may be null, X509 etc).

The primary function of these mechanisms is to provide for the mutual authentication of the communicating peers and to leverage confidentiality and integrity features at the transport layer.

As noted in the previous section on unilateral message authentication, bearer mechanisms do not necessarily provide message authentication and for this reason may be used in conjunction with mechanisms that do provide message authentication. In this case the bearer token MUST be used to determine the invoker identity for authorization decisions.

6.2.2.1. Processing Rules

These mechanisms MUST implement TLS/SSL end entity authentication in accordance with the TLS/SSL specifications and employing a cipher suite based on X.509 certificates, requiring the following:

- The sender MUST authenticate the recipient AND the recipient MUST authenticate the sender.
- The recipient MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.
- The sender MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.
Note that these X.509 certificates are those associated with SSL/TLS, and not necessarily associated with the WSS X.509 token profile.

### 6.3. Message Authentication

The non-null message authentication mechanisms prescribed by this specification generally rely upon the integrity properties obtained by using the OASIS standard SOAP Message Security mechanism in conjunction with a specified OASIS standard token profile. These mechanisms generally rely on the use of XML Signature technology as profiled by the OASIS specifications.

Message authentication mechanisms have distinct security properties regarding authenticity of a given message. For the mechanisms that include both peer entity authentication and message authentication, optimizations regarding attestation MAY be employed. For example, in environments where there is no requirement that a signature attesting to the authenticity of the message be retained, then it may be sufficient to rely upon the security properties of peer entity authentication to assure the integrity and authenticity of the message payload with no additional message layer signature.

The processing rules and requirements apply to all mechanisms used for Message Authentication where the token is bound to the message (i.e. this section does not apply to bearer tokens when they are not bound to the message). Additional requirements and processing rules may apply to a token as described for that specific token type, either in this specification or in a SecMech profile.

The message authentication mechanisms described in SecMech and its profiles are unilateral. That is, only the sender of the message is authenticated. It is not in the scope of this specification to suggest when response messages should be authenticated, but it is worth noting that the mechanisms defined in Section 6.4 could be relied upon to authenticate any response message as well. Deployers should recognize, however, that independent authentication of response messages does not provide the same message stream protection semantics as a mutual peer entity authentication mechanism.

### 6.3.1. Token Container

A token container type is defined to provide a uniform means to convey tokens, and allows a Web Services Security token to be directly contained the container, or to be referenced from the container. A reference may be an external reference to an token or a reference to another local token container.

The token container type (TokenType) is used to define elements in the ID-WSF namespace, including:

- InvokingIdentity header
- TargetIdentity header

In addition, a <Token> element is defined and should be used in the following locations:

- IdP Token Service Responses and in some cases inputs
- People Service Responses
- Liberty’s profile of the EPR in the Metadata SecurityContext element.
The following schema fragment describes the TokenType type and the corresponding <Token> element:

<!--
TokenType can refer to an external token using the ref attribute (no element content) or contain a Web Services Security token, or a WSS Security Token Reference (STR) element
-->

<xs:complexType name="TokenType">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional" />
  <xs:attribute name="ref" type="xs:anyURI" use="optional" />
</xs:complexType>

<xs:element name="Token" type="sec:TokenType" />

The following examples demonstrate the use of the <Token> element and the TokenType type:

• Token carrying a saml assertion:

  <Token id="x123">
    <saml2:Assertion id="x345" />
    ...
  </Token>

• Token referring to a Web Service Security token, either somewhere else in a message (local) or to an external token:

  <Token id="local-reference1" ref="#123" />
  ...
  <Token id="external-reference1" ref="http://somehost/gettoken" />

When an element of token container type (e.g. a <Token> element) references a <Token> element the reference MUST be to the <Token> element itself.

• Token carrying a Web Service Security security token reference (wsse:SecurityTokenReference) for an external token.

  A security token reference MUST only be used within an element of TokenType when that element is to be transmitted to a party as part of a web service message, and where that party will dereference the STR to locate the security token. A security token reference MUST only be an external reference.

  This reference would be used to support an "artifact"-like model, where the discovery service returns the STR in the EPR and which the WSC places the STR (without dereference) into the security header of the message to the WSP.

  <Token id="x678">
    <wsse:SecurityTokenReference wsu:ID="x789" wsse:TokenType="http://...#SAMLV2.0">
      <wsse:Reference URI="https://...?ID=x2323" />
    </wsse:SecurityTokenReference>
  </Token>
6.3.1. Common Sender Processing Rules

• The construction and decoration of the `<wsse:Security>` element MUST adhere to the rules specified in the [wss-sms].

• The sender MUST place the message authentication security token as a direct child of the `<wsse:Security>` element.

• For deployment settings which REQUIRE independent message authentication, the obligation MUST be accomplished by signing the message body and portions of the header and placing the `<ds:Signature>` element as a direct child of the `<wsse:Security>` header.

• For deployment settings which DO NOT REQUIRE independent message authentication then the subject confirmation obligation may be accomplished by correlating the certificate and key used to affect peer entity authentication with the certificate and key described by the message authentication token. To accommodate this, the assertion issuing authority MUST construct the assertion such that the confirmation key can be unambiguously verified to be the same certificate and key used in establishing peer entity authentication. This is necessary to mitigate the threat of a certificate substitution attack. It is RECOMMENDED that the certificate or certificate chain be bound to the subject confirmation key. If peer entity authentication is not used and the message is bound to SOAP with the Liberty SOAP Binding (v2.0) [LibertySOAPBinding], the sender MUST sign:

• The `<wsa:MessageID>` header block element.

• The `<wsu:Timestamp>` contained in the `<wsse:Security>` header block element.

• The `<wsa:RelatesTo>` header block element, when present.

• All other header block elements that require the aforementioned security properties in accordance with the security requirements prescribed in their respective specification.

If the message is signed the sender MUST include the resultant XML signature in a `<ds:Signature>` element as a child of the `<wsse:Security>` header.

The `<ds:Signature>` element MUST refer to the subject confirmation key with a `<ds:KeyInfo>` element. The `<ds:Keyinfo>` element MUST include a `<wsse:SecurityTokenReference>` element so that the subject confirmation key can be located within the `<wsse:Security>` header. The inclusion of the reference SHOULD adhere to the guidance specified in section 3.4.2 of [wss-saml11] (section 3.3.2 of [wss-saml]).

6.3.2. Common Recipient Processing Rules

• The recipient MUST locate the `<wsse:Security>` element for which it is the target. This MUST adhere to the rules specified in [wss-sms] and [wss-saml].

• The recipient MUST locate the security token and the recipient MUST determine that it trusts the authority which issued the token.

• The recipient MUST validate the issuer’s signature over the token. The recipient SHOULD validate the trust semantics of the signing key, as appropriate to the risk of incorrect authentication.
• If the message has been signed then the recipient MUST locate the `<ds:Signature>` element carried inside the `<wsse:Security>` header.

 The recipient MUST resolve the contents of the `<ds:KeyInfo>` element carried within the `<ds:Signature>` and use the key it describes for validating the signed elements.

 This validation MUST conform to the core validation rules described in [XMLDsig].

• If peer entity authentication is not in use and the message is bound with the Liberty SOAP Binding (v2.0) [LibertySOAPBinding], the recipient MUST verify the signature covers the following elements:

  • The `<wsa:MessageID>` header block element.
  • The `<wsu:Timestamp>` contained in the `<wsse:Security>` header block element.
  • The `<wsa:RelatesTo>` header block element, when present.
  • The `<S:Body>` element.
  • All other header block elements that require the aforementioned security properties in accordance with the security requirements prescribed in their respective specification.

6.4. WSS X.509 Token Authentication

The semantics and processing rules for mechanisms with MESSAGE having the value of X509 are described in this section. These URIs support unilateral (sender) message authentication and are of the form: urn:liberty:security:2003-08:PEER:X509 where PEER may vary depending on the peer authentication mechanism deployed (e.g. may be null, TLS etc).

The WSS X509 message authentication mechanism uses the Web Services Security X.509 Certificate Token Profile [wss-x509] as the means by which the message sender authenticates to the recipient. These message authentication mechanisms are unilateral. That is, only the sender of the message is authenticated. It is not in the scope of this specification to suggest when response messages should be authenticated but it is worth noting that this mechanism could be relied upon to authenticate the response message as well. Deployers should recognize, however, that independent authentication of response messages does not provide the same message stream protection semantics as a mutual peer entity authentication mechanism would offer.

For deployment settings that require message authentication independent of peer entity authentication, then the sending peer MUST perform message authentication by demonstrating proof of possession of the key associated with the X.509 token. This key MUST be recognized by the recipient as belonging to the sending peer.

When the sender wields the subject confirmation key to sign elements of the message the signature ensures the authenticity and integrity of the elements covered by the signature. However, this alone does not mitigate the threat of replay, insertion and certain classes of message modification attacks. To secure the message from such threats, one of the mechanisms which support peer entity authentication (see Section 6.2) MAY be used or the underlying SOAP binding request processing model MUST address these threats.

6.4.1. Sender Processing Rules

These rules are in addition to the generic message authentication processing rules specified in this document.
• The sender MUST demonstrate possession of the private key associated with the signature generated in conjunction
with the WSS X509 token profile.

For deployment settings which REQUIRE independent message authentication, the obligation MUST be accom-
plished by signing portions of the message as appropriate and recording information in the <wsse:Security>
header as outlined in [wss-sms].

For deployment settings which DO NOT REQUIRE independent message authentication then the sender MUST
accomplish this obligation by decorating the security header with a <ds:KeyInfo> element bearing the certificate.
This MUST be unambiguously verified to be the same certificate and key used in establishing peer entity
authentication. This is necessary to mitigate the threat of a certificate substitution attack. Also note that this
optimization only applies to ClientTLS:X509 mechanisms.

6.4.2. Recipient Processing Rules

• If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the
recipient MUST establish the correspondence of the certificate and key used to establish peer authentication with
the corresponding key information conveyed in the message. This allows the message recipient to determine
that the message sender intended a particular transport authenticated identity to be used. This information
may be conveyed in the message using an OASIS SOAP Message Security X.509 security token containing the
<ds:KeyInfo> MUST locate the <ds:Signature> element carried inside the <wsse:Security> header.

The recipient MUST resolve the contents of the <ds:KeyInfo> element carried within the <ds:Signature> and
use the key it describes for validating the signed elements. This validation MUST conform to the core validation
rules described in [XMLDsig]. Additionally, the recipient MUST determine that it trusts the key used to sign the
message, and the recipient SHOULD validate the sender's certificate, verifying the certificate revocation status as
appropriate to the risk of incorrect authentication.

6.4.3. X.509 v3 Message Authentication

The following example demonstrates the X.509 v3 message authentication mechanism.

<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sb="urn:liberty:lib:2005-11"
  xmlns:pp="urn:liberty:id-sis-pp:2003-08"
  xmlns:sec="urn:liberty:sec:2005-11"
  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"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <!-- see Liberty SOAP Binding Specification for which headers
    are required and optional -->
    <wsa:MessageID xmlns:id="mid">...</wsa:MessageID>
    <wsa:To>...</wsa:To>
    <wsa:Action>...</wsa:Action>
    <wsse:Security xmlns:wsse="">
      <wsu:Timestamp xmlns:wsu="">
        <wsu:Created>2005-06-17T04:49:17Z</wsu:Created>
6.5. Bearer Token Authentication

The Bearer mechanism is used to convey the authenticated identity of an invoker with a message. The mechanism is based on the presence of a bearer token in the security header of a message. A bearer token may include the endpoint reference for the discovery resource to which it applies, as well as the intended recipient of the assertion, so the scope of the assertion may be limited even though it is not bound to a specific message. In this situation, the bearer token is verified for authenticity and contributes to authorization decisions rather than being used to demonstrate the authenticity of the message.

The Bearer mechanism does not necessarily provide message authentication, since bearer tokens need not be bound to the message with a cryptographic signature. For this reason, if message authentication is desired a bearer mechanism may be used in conjunction with another mechanism used for message authentication, such as an X.509-based
mechanism. In this case the Bearer mechanism MUST be used to determine the invocation identity. (If the message
authentication identity differs, it may be assumed to be the sender, who may be different from the invoker).

Bearer token functionality may be implemented using different types of tokens, including tokens defined in OASIS
SOAP Message Security [wss-sms], such as WSS Binary Security Tokens (<wsse:BinarySecurityToken>), and
WSS Token profiles (X.509 token profile [wss-x509] or SAML token profiles [wss-saml11] for example). Custom
tokens or tokens which are subsequently profiled after this specification is finalized could still leverage the bearer
mechanism providing the wsse:ValueType is understood by the producer and consumer of the token. See the
example in Section 6.5.3.1.

The use of a bearer authentication mechanism is specified using a SecMech URN with a MESSAGE value of Bearer.
Such a bearer authentication mechanism supports unilateral (invoker) entity authentication. The URN is of the
form urn:liberty:security:2003-08:PEER:Bearer. PEER may vary depending on the peer authentication mechanism
deployed (e.g. may be null, TLS etc). Note that such URIs indicate that a bearer mechanism is in use, but do not
specify which exact specific bearer token instance is in use (e.g. SAML 2 assertion, binary security token, etc).

The type of bearer token must either be recognized from the schema of the token, as for example with a SAML
assertion, or from a ValueType attribute associated with the token, as for example with a WSS BinarySecurityToken.

This section defines normative requirements that apply in general to all bearer tokens. Additional detailed normative
requirements and semantics related to a specific bearer token type may be defined in a profile for that type. A profile
is not always required.

Specifically, the SecMech SAML Profile [LibertySecMech20SAML] defines additional normative requirements when
using SAML 2 assertions as bearer tokens. This core document provides normative requirements on the use of Binary
Security Tokens, see Section 6.5.3.

The following are general normative statements regarding the use of bearer tokens:

• A SAML 2 assertion may be used directly as a bearer token, when placed within a (<wsse:Security>) header
block. This usage is defined in the SecMech SAML profile [LibertySecMech20SAML].

• A bearer token MUST appear within the <wsse:Security> header of a message. That <wsse:Security>
header MUST be targeted at the recipient SOAP node to be used in authorization decisions by that entity.

• Note that the integrity, authenticity or confidentiality of the bearer token may not be protected when the bearer
token is neither signed nor encrypted at the message layer and secure end-to-end transport is not used. For this
reason caution must be taken not to expose the token to unauthorized entities.

To secure a message from such threats, one of the mechanisms which support peer entity authentication with
integrity and confidentiality protections (see Section 6.2) SHOULd be used in conjunction with or instead of an
unprotected bearer mechanism.

• The sender and receiver processing rules that follow must be followed.

6.5.1. Sender Processing Rules

• The construction and decoration of the <wsse:Security> header element MUST adhere to the rules specified in
[wss-sms].

• The sender MUST insert the bearer token as a direct child of the <wsse:Security> header and this header
MUST be targeted at the recipient.

6.5.2. Recipient Processing Rules
6.5.3. Binary Security Token Bearer Tokens

A bearer token MAY be a WSS Binary Security Token. The following normative requirements on the use of Binary Security Tokens as bearer tokens must be met:

- The EncodingType attribute MUST be explicitly stated to be the default, base64Binary.
- The ValueType MUST be present and indicate the format of the bearer token.

6.5.3.1. Custom Bearer Token Example (Informative)

This example depicts a custom security token being conveyed to the relying party. For such an example to function, the producer and consumer of the custom token must understand and follow the proper processing rules associated with the wsse:ValueType attribute.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:wsa="http://www.w3.org/2005/03/addressing"
 xmlns:sec="urn:liberty:sec:2005-11"
 xmlns:ss="urn:liberty:ss:2005-11"
 xmlns:pp="urn:liberty:id-sis-pp:2003-08"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-1.0.xsd"
>
<s:Header>
<!-- Custom binary security token -->
<wsse:BinarySecurityToken
 ValueType="anyNSPrefix:ServiceSessionContext"
 EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
>
<wsu:Id="id"/>
<wsu:Id="bst"/>
https://example.com/secureToken
<s:Body/>
</s:Envelope>
```
Liberty Alliance Project:
Liberty ID-WSF Security Mechanisms Core

QvAsufa9Vghnry5Blxe2VzwitMDwiRCS/bpbRQAFeBqmR2FyeSBGLiBFbGxpc29u
IDXnYXJ5lMvDbg1zb25ac3VuLnWvB6JARUBRABO25icfPHf179jM0DBArwBA/sG
YH+JpVmgR2EvJ0DyXz+s6yYMXZe4vPheBoFB7kA0UR+VapQ0qojJK+6ITyUx2w
R3egeJMPwqWpOA/EAk1lYE/xqcrqzddSqZsG43530/TT00Y+ENXxttVHbdJ79Rlk
8F8RJ3hLZjQjQbUZ7HRpY5EdA1jmltmQW/SGTKRz8uncs5BtmJjxkAbsku16Ys24E
Pv097/hV/Ut7tVW+8A/hkC6QV6ElUeVgPKwKzOph20LzuvwaFq9/TIN2RNGhCg
TNLvyuz+5yXGAY3rzn8UPe3NJRQexszBDpSpq3G07kL0VHN+/B/5GLSvcofpA
xj/JP+4lnsDgkYcWq1QeEeBABAgAJQBJ1+DoxWwhhAhBaAoJEPCJEJL9ultFsMqHg
9AiIZbumPkvx3DQCqcz+2Rjasy2YyuuSkWp9j7snFfwvBGTsA+2w03uLn8A596w
n4MVHxht5S2c2zMDKZABJBo8Qbttb51tQa1JMPq71gjmNjzazebqPfwpQHqJ66cje
de/3QbxBD/rXVYzsiEcdEQqRwbauC90c3TonrJBoP6h6sJkjGvQeJvjuutikMN
A97o00C1JKN1lEjVW14zwf7cmHwWyfC6418pqMEfLC7X7rYe7pXALZ6p1i8Ta5njGL
ldWryWwSDMCEuOt5wu1uYz+8Xvy1jXk1Rn156ioT5UHxGJq6q40ZONDWMdIhw
s19v1kuHhJuWz8BDzI0117qG2RfySFBxgp29uDxmnwAaW50ZXJoWNRmL51
Ad63ARQDBRA+dLR81Kkv260UBAxBse/URCD8wayj9v7qMKX391dxk/K316yj
moQSmsFkOsLD6Ej3iwptcR90eTo2F/5tJ0K9SFNaeffFupjVg97y+1DHHVqYqW
kDQg7YB5+kfs1sEubpU1mYvenhmgrU2Lnm0BDpby+UukRGjRLtDsuEUNX8fpGb27d
ddodor3kInl90pRPGQ/Flmduat28M9MFn4RpOw8N7PlxvPNsTXgGfFLY2DDO
Dk5he7RsA3rJl90f01i9nLH01XihwWFX71e66yv1HClAIlNwP08UXe1gkD3A
2fZMFMUx6yTzd90l1+ByDk/jLsGvs26t1RGSvW5SW00crBpKa3sVmsMyOC2dmZUBz
dW4uY29tI/QETAVuqP3jIwvCJEJL9ultFAQDGRdmpwmhrq1rACyA2a2yFoex0giz
NrtQvM1Rw5EzouGu9NMQ5Ms1sBpIMccA6LYY6rbQzqrP7P0U208uuQA1fPzRs
i41hsZOD0eKKAiW7G3bo0+dXdpwFPH7C7PObof8sJ71WBO+0BFK9Q8J3ZfYbGKc
tEO6KvO3fVWHhE12ehv52O787J0GbdN34P5NcLeRe69eLFJP3XN0nLQMVWmHm
0k5lsr1cRrPAPyWkG6vky/QVZeZSVPma9t43XlJUc+yDTsDvtrMmWwHUJ3/Pn8tIcA
XsoCvX/5Fup0ntpxOA1lUp0cCGpGzv1q7TmFsF5aSp01Om79q010/S060Gdkh
cyNqXWxsXvmbAI8Z2Z1QH01b15j2b0+1qEUXaUQW3pmAVCJEJL9ultFAQW1tw0F
CAY7BNB7n4+wMyH5HsUXFrQ2R1vs5DjzjK0x7j6pHDQhjfs24NLvvpufzA
uTE27FDIh+C05K5cGUTQx/xnkMm+HM87vPcChb3S1TGT+yxyVIqyI9Bb15nX2
Q19Rks3ZD0Xux3zu0DNX7dyKNX6fYkXKGRserWHxdILppnmvLodKCC/3zkKq9fz
VTr4y9tpfVblue1t9V3XGBU4ezcScDm9e+IEp+pQjnv9r9ac1nW5k/CJE0UJhh
oGfjyIm0Lutezhzw/g0eLVtkywsMgDr77qWXzRw0V1w0lqotUdTeceuRBIDAN+KvZ
vLR1tC3AaAGUNIjtkLDg1ti

</wsse:BinarySecurityToken>

<!-- this is the reference to the above bearer token -->

<wsse:SecurityTokenReference>

<wsse:Reference URI="#bst"/>

</wsse:SecurityTokenReference>

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

<ds:SignedInfo>

<ds:Reference URI="mid">
<!-- include the MessageID in the signature -->
</ds:Reference>

<ds:Reference URI="tid">
<!-- include the Timestamp in the signature-->
</ds:Reference>

<ds:Reference URI="#MsgBody">
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
</ds:Reference>

</ds:SignedInfo>

...</ds:Signature>
</s:Body>
</s:Envelope>
6.6. Identity Tokens

Identity Tokens are references to a principal that differ from an Authentication Token in that the Identity Token is primarily used to convey an identity while an Authentication Token conveys both the Identity and the authentication context of the user.

It is possible to use an Authentication token in the context where an Identity Token is needed (although the reverse is not appropriate), but there are differences that should be considered:

- Identity tokens typically are long lived since they don’t authenticate a user.
- Identity tokens represent a handle to be used to refer to the principal when the principal is not involved in a transaction (such as when Bob attempts to view Alice’s pictures – Alice may not even be logged in, but Bob may need a handle to pass to Alice’s picture WSP so that the WSP knows who’s pictures are being accessed).

Different mechanisms may be used to convey an identity token.

- A SAML 2.0 assertion may be used as an identity token. This usage is defined in the SecMech SAML profile [LibertySecMech20SAML].
- A WSS Binary Security Token may also be used as an identity token, if it has the appropriate ValueType attribute definition.
- A WSS SecurityTokenReference element may also be used to reference an identity token.
- Other XML definitions may also be possible.

Any identity token SHOULD be able to convey information needed for discovery. This is typically an endpoint reference.

An identity token must have an attribute of type IDType that may be used as a target of a ds:Reference, e.g. an xml:id or wsu:Id attribute.

Normative details using SAML 2 assertions are given in the Security Mechanisms SAML profile [Liberty-SecMech20SAML].
7. Message Authorization Mechanisms

The Message Authorization Model specifies OPTIONAL mechanisms to convey authorization and resource access information (supplied by a trusted third party) that may be necessary to access a service. This facility, incorporated for authorization purposes, serves a distinct and complementary function to the binding between subject and key that the subject accomplishes for authentication purposes. However, it is possible to optimize the processing when the message authentication mechanism utilizes the same subject confirmation key as the authorization mechanism and the key has successfully been applied to ensure the integrity and authenticity of the message payload.

7.1. Authorization Mechanism Overview (Informative)

The authorization mechanism defined by this specification formalizes the generation and conveyance of authorization information. In support of this mechanism a Trusted Third Party (TTP) may be relied upon to act as either a Policy Information Point (PIP), a Policy Decision Point (PDP) and potentially a coarse grained Policy Enforcement Point (PEP). As a PIP the authority may provide information useful in making a policy decision to the relying party. As a PDP, the Trusted Third Party may make coarse access decisions, such as during the discovery process disallowing discovery of a resource if not authorized. This requires strong assurance as to the authenticity of a peer subject. Given the reliance of authorization upon authentication, this model aids in disseminating subject confirmation obligations, identity information and access authorization data.

The authorization model supports the issuance of assertions that convey information regarding the resource to be accessed, the entity attempting to access the resource, the mechanism by which the accessing entity must use to confirm its identity to the recipient and the ability for the sending entity to access the resource on behalf of another system entity.

When one provider acts on behalf of an invoker, information about both the sender and invoker may be useful for a subsequent authorization decision and may need to be conveyed with the message, including information needed to verify both identities.

7.2. Authorization Assertion Generation

The Liberty Alliance Discovery service, [LibertyDisco], is a trusted service which enables the discovery of identity-based web services. The trusted authority [LibertyDisco] may issue an assertion, subsequently used when accessing the discovered identity-based web service (the resource).

In addition to managing the registration and discovery of identity-based web services the trusted authority may act as a centralized policy information and decision point. The authority may issue assertions regarding authentication and authorization policies enforced for a given identity-based web service, resource and the identity of the sender. The makeup of this assertion reflects the information necessary to accommodate the authentication and authorization policy requirements.

Specific processing rules are provided in the SecMech SAML 2 profile.

7.3. Provider Chaining

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. This mechanism allows proxying to be performed, where each provider proxies the request to the next party. An example is a browser client accessing a portal which acts as a web service client on behalf of the browser client, accessing a web service provider that in turn passes the request to a second web service provider. 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 provider to make an appropriate authorization decision, since knowledge of the sender may not be enough information.
A security token is passed to each service provider at each step, and ultimately to the destination service provider. This token contains authentication and authorization information about the principal invoking the service. This principal is the subject of the assertion. This token is also passed to the assertion issuer by each WSP that wishes to pass the request on to another WSP, allowing the assertion issuer to issue a modified token appropriate to the next WSP.

When two or more WSPs are transited before reaching the destination WSP, a `<ProviderTransitedStatement>` SHOULD be included in the assertion. The normative details of how this is to be done using SAML 2 assertions is provided in the Security Mechanisms SAML profile [LibertySecMech20SAML].

The `<ProviderTransitedStatement>` SHOULD capture the identity of all but the last transited provider. For example, if there were three WSPs transited before reaching the final (fourth) WSP, it is only the first two that are recorded in the `<ProviderTransitedStatement>`. To be meaningful to an authorization decision, the provider path MUST be recorded by a trusted party. In this case the trusted party is the Discovery Service that issues the token.

The last transited provider need not be explicitly recorded in the `<ProviderTransitedStatement>` since it is known to the message recipient as the sender of the message. The identity of the last transited provider MUST be recorded in the assertion, however, for example as part of the SAML assertion confirmation method.

This information may be used by a policy decision point to make an authorization decision to determine whether the service provider offers the service. This decision point may be the service provider itself or some other system entity - how authorization is implemented is outside the scope of this specification, apart from the information conveyed in the message to enable such decisions.

The following table gives an example of the information contained in a token as it traverses a number of providers. This shows the system entities (A-F) where A is assumed to be a web browser client, and B-F are WSPs. B-E also act as WSCs and F the destination WSP.

<table>
<thead>
<tr>
<th>Party:</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assertion Contains:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject = principal = invoker</td>
<td>A(v)</td>
<td>A(w)</td>
<td>A(x)</td>
<td>A(y)</td>
<td>A(z)</td>
<td></td>
</tr>
<tr>
<td>sender(assertion confirmation method)</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider Chain</td>
<td></td>
<td>(B)</td>
<td>(B,C)</td>
<td>(B,C,D)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each entry of this table shows the relevant content of the assertion as received by the party at the top of that column. Thus, for example, WSP E receives an assertion showing that the invoker is A and that the sender is D. WSP E also receives a provider chain showing that providers B and C were transited before the request reached D. Note that each WSP may receive name identifiers that are unique to it and the sender, for example "y" instead of "A" for the invoker, and possibly other name identifiers for the sender and provider chain than other WSPs would receive.

It is the responsibility of each transited provider to present the information needed by the discovery service to create the token for the next step. When a WSP receives a request and determines that it must act as a WSC to send the request to another WSP, it looks for the presence of any `<ProviderTransitedStatement>` in the Assertion of the request and, if present, sends the `<ProviderTransitedStatement>` up to the assertion issuer after adding the WSC from which it just received the request to the `<ProviderTransitedStatement>`. If there is no `<ProviderTransitedStatement>` in the request, the WSP creates a `<ProviderTransitedStatement>` containing the WSC from which it just received the request.

The assertion issuer, upon receiving the request from the WSP, verifies that further transiting of providers is allowed, and, if so, creates a new assertion for the WSP (soon to be a WSC) with the new `<ProviderTransitedStatement>`. The assertion issuer may map the name identifier of the assertion subject as well as for the members of the provider
chain, to name identifiers appropriate for the current WSP (soon to be WSC) and the next WSP. This is done to protect privacy.

When the WSP receives the new assertion from the assertion issuer, it sends it on to the recipient, which may be the destination WSP or a WSP that may act as a WSC to send the request to another WSP, repeating the process. Although the token issued by the discovery service has a name identifier for the same principal as the subject of the original assertion, the name identifier may be changed to maintain privacy. This token also contains the revised `<ProviderTransitedStatement>`. Because of these changes, each token is in fact a new token, with a new Discovery Service signature.

When a WSP acts as a WSC to send a request to the next WSP, it is the sender. Again, this sender identity may be expressed using a name identifier. The sender’s identity is conveyed as part of the subject confirmation method, which includes the name identifier for the sender. This may use various confirmation methods, including sender-vouches, holder-of-key and bearer.

When a `<ProviderTransitedStatement>` is used, a single `<ProviderTransitedStatement>` statement MUST be used to contain the information about all of the transited WSPs, in a single statement. (In earlier versions of ID-WSF, Security Mechanisms 1.2 and earlier [LibertySecMech12], the chain was expressed by a separate `<ProxyTransitedStatement>` for each proxy transited.)

When a `<ProviderTransitedStatement>` is included in a token, it contains `<SubjectConfirmation>` elements to indicate the identity of each transited WSP to the recipient. The ProviderConfirmationData elements within the `<ProviderTransitedStatement>` include a reference to the Assertion that the proxy received before obtaining a new assertion from the Discovery Service. These `<AssertionIDRef>` elements allow the assertion issuer to track down the original assertion. Normative details are defined in the SecMech SAML profile [LibertySecMech20SAML].

When requesting a token from the assertion provider, the WSP acting as a transited provider SHOULD convey its confirmation claim in the form of a SAML assertion carried as a security token within the security header of the request to the assertion issuing authority when requesting a token.

The basic scenario is that each access to the discovery service will provide each WSP with the endpoint reference to allow the next WSP to access the appropriate discovery service in the next step of transiting the WSPs. In addition, at each step the discovery service will provide an appropriate security token to the WSP to allow the next WSP to be able to access the discovery service before acting as a WSC and make the next request. This continues until the final step where the token is presented to the service provider. Note that including necessary information in each Discovery Service request eliminates the need for the Discovery Service to maintain state information.

The final service provider may make an authorization decision based on the information presented to it in the request, as well as information it knows. Including information about a transited WSP path may be useful to this authorization decision.

Various tokens may be used to convey provider chaining information. SAML 2.0 assertions SHOULD be used. How SAML 2.0 assertions are to be used is outlined in the Security Mechanisms SAML profile [LibertySecMech20SAML].

### 7.3.1. Supporting Schema

#### 7.3.1.1. ProviderTransitedStatement Schema

The `<ProviderTransitedStatement>` is used to identify the WSPs that are transited, apart from the last WSP that is transited. The intended usage of the statement is twofold:

- First, provide the authorization decision point associated with the final service provider transited WSP path information necessary to make an authorization decision.

- Second, carry the transited path state on behalf of the assertion issuer.
The following schema fragment describes the structure of the `<ProviderTransitedStatement>` element.

```xml
<xs:element name="ProviderTransitedStatement" type="sec:ProviderTransitedStatementType"/>
<xs:complexType name="ProviderTransitedStatementType">
  <xs:extension base="saml2:StatementAbstractType">
    <xs:sequence>
      <xs:element ref="saml2:SubjectConfirmation" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexType>
```

7.3.1.2. ProviderInfoConfirmationData Schema

A transited provider uses the `<ProviderInfoConfirmationData>` to supply subject confirmation data to an assertion issuer; this subject confirmation data previously was used by another WSP in authenticating a message sent to the first provider.

The following schema fragment describes the structure of the `<ProviderInfoConfirmationData>` element.

```xml
<!DOCTYPE --
<xs:complexType name="ProviderInfoConfirmationDataType" mixed="false">
  <xs:complexContent>
    <xs:restriction base="saml2:SubjectConfirmationDataType">
      <xs:sequence>
        <xs:element ref="saml2:AssertionIDRef"/>
        <xs:element ref="saml2:Issuer"/>
        <xs:element name="IssueInstant" type="xs:dateTime"/>
        <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute ref="xml:id" use="optional"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

The semantics around the elements are as follows:

- The `<saml2:AssertionIDRef>`, `<Issuer>` and `<IssueInstant>` are that of the `<saml2:Assertion>` presented by the proxy subject.

- The OPTIONAL `<ds:Signature>` element is a digital signature created by the recipient which covers the child elements of `<ProviderInfoConfirmationData>` with the exclusion of itself. It is RECOMMENDED that the enveloped signature transform (see [XMLDsig]) be utilized to accomplish the element exclusion.
7.4. Presenting Authorization Data

Interactions with identity-based web services may rely on the conveyance of authorization information. In general, a trusted authority issues the authorization data. In such a setting the authorization information would be sent along with the identity-based web service request to the recipient. See Authorization Assertion Generation (Section 7.2) for details as to how this data is acquired and formulated.

7.4.1. Processing Rules

- The sender MUST authenticate to the recipient using one of the authentication mechanisms described in Message Authentication (Section 6.3).

It is RECOMMENDED that the sender authenticate using SAML assertion message authentication and specifically conform to the processing rules specified in the SecMech SAML 2 profile.

7.5. Consuming Authorization Data

A recipient that exposes a resource typically makes access control decisions based on the invocation identity. Additionally the recipient may also predicate access control policies upon the sender identity. The semantics of resource access authorization are described in Presenting Authorization Data (Section 7.4).

Additional details related to the use of SAML 2.0 assertions are presented in the SecMech SAML 2.0 profile.

7.5.1. Processing Rules

- The recipient MUST authenticate the sender using one of the mechanisms described in Authentication Mechanisms.

Additional processing rules specific to the use of SAML 2.0 assertions are presented in the SecMech SAML 2 profile.
8. Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:sec:2005-11"
  xmlns:xenc="http://www.w3.org/2001/04/xmllenc#"
  xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:disco="urn:liberty:disco:2005-11"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sec="urn:liberty:sec:2005-11"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:md="urn:liberty:metadata:2004-12"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
    schemaLocation="saml-schema-assertion-2.0.xsd"/>
  <xs:import namespace="urn:liberty:disco:2005-11"
    schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
  <!-- <xs:import namespace="urn:liberty:ac:2005-11"
    schemaLocation="liberty-authentication-context-v2.0.xsd"/> -->
  <xs:import namespace="urn:liberty:metadata:2004-12"
    schemaLocation="liberty-metadata-v2.0.xsd"/>
  <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
  <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    schemaLocation="wss-secext-1.0.xsd"/>
  <!-- <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    schemaLocation="wss-utility-1.0.xsd"/> -->
  <xs:annotation>
    <xs:documentation>
      The source code in this XSD file was excerpted verbatim from:
      Liberty ID-WSF Security Mechanisms Specification
      Version 2.0-12
      21 September 2005
      Copyright (c) 2005 Liberty Alliance participants, see
      http://www.projectliberty.org/specs/idwsf_2_0_r2_copyrights.php
    </xs:documentation>
  </xs:annotation>
</xs:schema>
```

Liberty Alliance Project

36
<xs:element name="ProviderTransitedStatement" type="sec:ProviderTransitedStatementType"/>
<xs:complexType name="ProviderTransitedStatementType">
  <xs:extension base="saml2:StatementAbstractType">
    <xs:sequence>
      <xs:element ref="saml2:SubjectConfirmation" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexType>

<!-- TokenType can refer to an external token using the ref attribute (no element content) or contain a Web Services Security token, or a WSS Security Token Reference (STR) element -->
<xs:complexType name="TokenType">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional" />
  <xs:attribute name="ref" type="xs:anyURI" use="optional" />
</xs:complexType>
<xs:element name="Token" type="sec:TokenType"/>
</xs:schema>
References

Normative


