Liberty ID-WSF Security Mechanisms Core

Version:

2.0-errata-v1.0

Editors:
Frederick Hirsch, Nokia Corporation
Contributors:
Robert Aarts, Hewlett-Packard
Conor Cahill, Intel Corporation, formerly America Online, Inc.
Carolina Canales-Valenzuela, Ericsson
Scott Cantor, Internet2, The Ohio State University
Darryl Champagne, IEEE-ISTO
Gary Ellison, Sun Microsystems, Inc.
Jeff Hodges, Neustar
John Kemp, Nokia Corporation
John Linn, RSA Security Inc.

Rob

Lockhart
, IEEE-ISTO
Paul Madsen, NTT, formerly Entrust
Jonathan Sergent, Sun Microsystems, Inc.
Greg Whitehead, Hewlett-Packard

Abstract:

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

Filename: liberty-idwsf-security-mechanisms-core-2.0-diff-v1.0.pdf

Notice

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

Contents

1. Introduction

This document specifies security mechanisms for identity-based web services. This includes mechanisms for authentication, integrity and confidentiality protection, and the means for sharing information necessary for authorization decisions. The mechanisms build on accepted technologies including SSL/TLS, XML-Signature [XMLDsig] and XML-Encryption [xmlenc-core], and SAML assertions. OASIS Web Services Security SOAP Message Security [wss-sms11] compliant header elements are used for message level security, to communicate the relevant security information, for example using SAML [SAMLCore11] or [SAMLCore2] assertions, along with the protected message. A separate SAML Security Mechanism profile is defined for the use of SAML security tokens in conjunction with this core document [LibertySecMech20SAML].

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 requesting 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 [Schema1-2]) 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 [RFC2119].

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

Prefix	Namespace
sec:	`urn:liberty:security:2006-08` This namespace is used for Liberty ID-WSF 2.0 Security Mechanisms.
sb:	`urn:liberty:sb:2006-08` This namespace represents the Liberty SOAP Binding namespace (v2.0). It is defined in the Liberty SOAP Binding document, v2.0 [LibertySOAPBinding].
disco:	`urn:liberty:disco:2006-08` This namespace represents the Liberty discovery service. It is defined in [LibertyDisco].
saml:	`urn:oasis:names:tc:SAML:1.0:assertion` This namespace represents SAML 1.0 assertions. It is defined in [SAMLCore11].
saml2:	`urn:oasis:names:tc:SAML:2.0:assertion` The prefix `saml2:` stands~~namespace~~ for the~~represents~~ SAML v2 assertion~~2.0~~ namespace. It is defined in [SAMLCore2].
samlp2:	`urn:oasis:names:tc:SAML:2.0:protocol` The prefix `samlp2:` stands for the SAML v2~~assertions.~~ protocol namespace. It is defined in [SAMLCore2].

Prefix	Namespace
S:	`http://www.w3.org/2002/12/soap-envelope` This namespace represents the SOAP 1.2 namespace. It is defined in [SOAPv1.2].
ds:	`http://www.w3.org/2000/09/xmldsig#` This namespace represents the XML Signature namespace. It is defined in [XMLDsig].
xenc:	`http://www.w3.org/2001/04/xmlenc#` This namespace represents the XML Encryption namespace. It is defined in [xmlenc-core].
wsa:	`http://www.w3.org/2005/08/addressing` This namespace represents the WS-Addressing namespace. It is defined in [WSAv1.0].
wsse:	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd` This namespace represents the SOAP Message Security namespace. It is defined in [wss-sms11].
wsse11:	`http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-wssecurity-secext-1.1.xsd` This namespace represents the SOAP Message Security v1.1 namespace. It is defined in [wss-sms11].
wsu:	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd` This namespace represents the SOAP Message Security Utility namespace. It is defined in [wss-sms11].
xs:	`http://www.w3.org/2001/XMLSchema` This namespace represents the W3C XML schema namespace. It is defined in [Schema1-2].
xsi:	`http://www.w3.org/2001/XMLSchema-instance` This namespace represents the XML Schema instance namespace. It is defined in [Schema1-2].

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 derived from the usage scenarios.

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 met.

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 (see [RFC4346]) 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 Web Services Security (WS-Security) 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 where 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 so that applicable privacy regulations are followed. 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 and privacy 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-sms11] 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 SHOULD 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 Security Mechanisms SAML profile describes how to accomplish this.

6. Authentication and Integrity Mechanisms

This specification defines a set of authentication and integrity mechanisms, labeled by URIs, to support various security requirements. Multiple mechanisms are specified 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

These mechanisms may provide integrity, confidentiality and authentication, but the peer mechanism does not provide end to end integrity or confidentiality in the presence of SOAP intermediaries.

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 2. Authentication Mechanism Versions

DATE	ID-WSF version
2003-08	ID-WSF 1.0
2004-04	ID-WSF 1.0 Errata
2005-02	ID-WSF 1.1
2006-08	ID-WSF 2.0

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 3. Peer Authentication Mechanisms

PEER	Mechanism
null	None
TLS	Peer recipient (SSL/TLS server) authentication
ClientTLS	Mutual Peer authentication

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 4. Message Authentication Mechanisms

MESSAGE	Mechanism
null	None
SAML	Use of SAML 1.x assertions in conjunction with SOAP Message Security, as outlined in earlier versions of the Security Mechanisms specification.
SAMLV2	Use of SAML 2.0 assertions in conjunction with SOAP Message Security, as outlined in the Security Mechanisms SAML profile.
X509	SOAP Message Security X509 Token Profile invoker authentication
Bearer	Bearer token invoker authentication
peerSAMLV2	Use of SAML 2.0 assertions in conjunction with SOAP Message Security, with a PEER layer key as the confirmation key, for example the client SSL/TLS key. This mechanism is intended to be used when the message is not signed.

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.

When SAML assertions are used for the SAMLV2, peerSAMLV2 or Bearer MESSAGE mechanisms, the following SAML 2.0 Confirmation Method attribute values correspond to the Security Mechanism identifiers:

Table 5. Confirmation Methods for Mechanisms using SAML 2.0

MESSAGE	SAML 2.0 Confirmation Method
SAMLV2	urn:oasis:names:tc:SAML:2.0:cm:holder-of-key
Bearer	urn:oasis:names:tc:SAML:2.0:cm:bearer
peerSAMLV2	urn:oasis:names:tc:SAML:2.0:cm:holder-of-key

The following table summarizes 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 6. Authentication Mechanisms

Mechanism	Peer Entity	Message
urn:liberty:security:2003-08:null:null	No	No
urn:liberty:security:2005-02:null:X509	No	Yes
urn:liberty:security:2005-02:null:SAML	No	Yes
urn:liberty:security:2006-08:null:SAMLV2	No	Yes
urn:liberty:security:2005-02:null:Bearer	No	Yes [1]
urn:liberty:security:2003-08:TLS:null	Recipient	No
urn:liberty:security:2005-02:TLS:X509	Recipient	Yes
urn:liberty:security:2005-02:TLS:SAML	Recipient	Yes
urn:liberty:security:2006-08:TLS:SAMLV2	Recipient	Yes
urn:liberty:security:2005-02:TLS:Bearer	Recipient	Yes [2]
urn:liberty:security:2003-08:ClientTLS:null	Mutual	No
urn:liberty:security:2005-02:ClientTLS:X509	Mutual	Yes
urn:liberty:security:2005-02:ClientTLS:SAML	Mutual	Yes
urn:liberty:security:2006-08:ClientTLS:SAMLV2	Mutual	Yes
urn:liberty:security:2005-02:ClientTLS:Bearer	Mutual	Yes [2]
urn:liberty:security:2006-08:ClientTLS:peerSAMLV2	Mutual	Yes [3]

[1] The bearer token is not bound to the message and is not protected by the TLS mechanism in this case.

[2] The bearer token is not bound to the message at the SOAP Message layer. It is integrity and confidentiality protected by TLS for a single TLS link, assuming correct ciphersuite use, but not protected end-end if the SOAP message traverses SOAP intermediaries.

[3] The SSL/TLS client key is also the message confirmation key in this case. This means the key~~to~~ need not be expected within ~~determine~~ the SOAP message conveyed as part of SOAP Message security~~key~~ when this Security Mechanism~~Mechanisms URI~~ is specified and used.~~known.~~

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 [RFC4346]. 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 profile [LibertySecMech20SAML]) 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 with a signature.

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 token 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 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:2006-08: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 and Integrity

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. SSL/TLS transport level security is designed to provide integrity protection in conjunction with authentication. Note that peer authentication may not provide adequate integrity, confidentiality or authentication 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 Channel Protection (Section 5.1).

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 and Integrity

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 WSS X.509 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 in the container, or to be referenced from the container. A reference may be an external reference to a token or a reference to another local token container.

The token container type (TokenType) may be ~~is~~ used to define elements in the ID-WSF namespace, and~~including the following~~ has~~elements: InvokingIdentity~~ also been~~element TargetIdentity~~ used to~~element In~~ define a <Token> element in the security mechanisms namespace. This <sec:Token>~~and~~ element may~~should~~ be used in a number of ID-WSF 2.0 schema definitions, such as: ~~locations:~~

- The~~IdP~~ security context container type used in the Discovery Service to profile EPRs, ~~cases inputs~~
- The~~People~~ mapping input and output types~~Service~~ for the Identity Mapping Service, and ~~Responses~~
- The~~Liberty's profile of the EPR~~ AddKnownEntityRequestType for~~in~~ the People~~Metadata SecurityContext~~ Service. ~~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:attribute name="usage" type="xs:anyURI" use="optional" />
</xs:complexType>

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

This specification defines the following URN values for the `usage` attribute (others may be defined elsewhere):

- ~~`urn:liberty:security:tokenusage:2006-08:InvocationIdentity`~~ `urn:liberty:security:tokenusage:2006-08:TargetIdentity`
- `urn:liberty:security:tokenusage:2006-08:SecurityToken`

These~~In~~ two URNs are used when the token is contained~~the token~~ in an EPR to~~would~~ be used to create a~~the corresponding~~ SOAP header by the Discovery Service. The TargetIdentity usage indicates that~~InvocationIdentity would~~ the token should be used to create an `<sb:TargetIdentity>`~~Invocation Identity header,~~ header block. Any~~the TargetIdentity~~ token with the SecurityToken usage in an~~a SecurityToken~~ EPR is placed in a `<wsse:Security>` header block.

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" ...>
    ...
  </saml2:Assertion>
</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>
```

6.3.2. Message Integrity rules for senders and receivers

This section only applies if SOAP message security is used for a message bound to SOAP (i.e., is a "SOAP-bound-ID-* message") according to the Liberty SOAP Binding (v2.0) [LibertySOAPBinding].

In this case the sender MUST create a single `<ds:Signature>` contained in the `<wsse:Security>` header and this signature MUST reference all of the message components required to be signed.

In particular, this signature MUST reference the SOAP Body element (the element itself), the security token associated with the signature, and all headers in the message that have been defined in the Liberty SOAP Bindings specification, including both required and optional header blocks [LibertySOAPBinding].

An example security token is a `<saml2:Assertion>` element conveyed in the `<wsse:Security>` header.

The wsu:Timestamp header in the wsse:Security header block, the wsa:MessageID, wsa:RelatesTo, sb:Framework, sb:Sender and sb:InvocationIdentity header blocks are examples of header elements that would be referenced in a signature.

Note that care must be taken when constructing elements contained in Reference Parameters in Endpoint References, as these will be promoted to SOAP header blocks. Effort should be taken to avoid conflicting or duplicate id attributes, for example by using techniques to generate ids where it is highly likely that they are unique.

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]).

6.3.3. Common Sender Processing Rules

- The construction and decoration of the `<wsse:Security>` header element MUST adhere to the rules specified in the [wss-sms11].
- The `<wsse:Security>` header element MUST have a `mustUnderstand` attribute with logical value `true`.
- The sender MUST place the message authentication security token as a direct child of the `<wsse:Security>` element.
- The sender MUST follow the message integrity rules outlined in the previous section Message Integrity rules for senders and receivers (Section 6.3.2) when message authentication mechanisms are used.

The following considerations do not apply to Bearer tokens:

- 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>` 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.

6.3.4. 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 [wss-sms11] and the applicable WSS token profiles (e.g., [wss-saml] for SAML tokens).

- The <wsse:Security> header element MUST have a mustUnderstand attribute with logical value true and the recipient must be able to process this header block according to WSS [wss-sms11] and the appropriate WSS token profiles (e.g., for SAML the SAML token profile [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. This validation MUST conform to the core validation rules described in [XMLDsig]. 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.

  Unless the security mechanism is peerSAMLV2 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. When the security mechanism is peerSAMLV2 the key is the client key used in SSL/TLS client authentication.

- The sender MUST follow the message integrity rules outlined in the previous section Message Integrity rules for senders and receivers (Section 6.3.2) when message authentication mechanisms are used.

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 accomplished by signing portions of the message as appropriate and recording information in the `<wsse:Security>` header as outlined in [wss-sms11].

  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 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. Information relating the SSL/TLS key to the message MAY be conveyed in the message using an OASIS SOAP Message Security X.509 security token.

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:sb:2006-08"
    xmlns:pp="urn:liberty:id-sis-pp:2003-08"
    xmlns:sec="urn:liberty:security:2006-08"
    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 wsu:Id="mid">...</wsa:MessageID>

    <wsa:To wsu:Id="to">...</wsa:To>

    <wsa:Action wsu:Id="action">...</wsa:Action>

    <wsse:Security mustUnderstand="1">

      <wsu:Timestamp wsu:Id="ts">
        <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
      </wsu:Timestamp>

      <wsse:BinarySecurityToken
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
          wsu:Id="X509Token"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64
          MIIB9zCCAWSgAwIBAgIQ...
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>

          <!-- in general include a ds:Reference for each wsa: header
               added according to SOAP binding -->

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

          <!-- include the To in the signature -->
          <ds:Reference URI="#to">...</ds:Reference>

          <!-- include the Action in the signature -->
          <ds:Reference URI="#action">...</ds:Reference>

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

          <!-- bind the security token (thwart cert substitution attacks) -->
          <ds:Reference URI="#X509Token">
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>Ru4cAfeBABE...</ds:DigestValue>
          </ds:Reference>

          <!-- bind the body of the message -->
          <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:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#X509Token" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <ds:SignatureValue>
          HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
        </ds:SignatureValue>
      </ds:Signature>
    </wsse:Security>
  </s:Header>
  <s:Body wsu:Id="MsgBody">
    <pp:Modify>
      <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
  </s:Body>
</s:Envelope>
```

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-sms11], 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 Custom Bearer Token example (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 observed.

6.5.1. Sender Processing Rules

- The construction and decoration of the `<wsse:Security>` header element MUST adhere to the rules specified in [wss-sms11].

- 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

- The recipient MUST locate the `<wsse:Security>` element for which it is the SOAP target. This header MUST adhere to the syntax and processing rules specified in [wss-sms11].

- The recipient MUST locate the bearer token by locating it as a direct child of the appropriate `<wsse:Security>` header. The recipient can recognize the token by ValueType in the case of a Binary Security Token, or by using its well known schema type.

- The recipient MUST process the token in accordance with the processing rules of the token type, as indicated by its schema and namespace.

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 `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 version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:sb="urn:liberty:sb:2006-08"
    xmlns:pp="urn:liberty:id-sis-pp:2003-08"
    xmlns:sec="urn:liberty:security:2006-08"
    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/2005/03/addressing">

  <s:Header>
    <!-- see Liberty SOAP Binding Specification for which headers
         are required and optional -->

    <wsa:MessageID wsu:Id="mid">...</wsa:MessageID>

    <wsa:To wsu:Id="to">...</wsa:To>

    <wsa:Action wsu:Id="action">...</wsa:Action>

    <wsse:Security mustUnderstand="1">

      <wsu:Timestamp wsu:Id="ts">
        <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
      </wsu:Timestamp>

      <!-- 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="bst" >
        mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
        YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpM+sF2BnpND118f/mXCv3XbWhiL
        xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
        QXdp1/HIYQo5tIvCAM4pGk8nJFh6JrLsOEnT887aJRaasvBAAQ27C7D4Dmpt01aC
        FqLEQ98/lt6nkFmf7oiuZkID++xQXn74LWOvdNlki43VaSXWcQAjzCzirHSuVX1N
        QvAsufa9Vghnry5Blxe2VzwitMDwiRCS/bpbRQAFEbQmR2FyeSBGLiBFbGxpc29u
        IDxnYXJ5LmVsbGlzb25Ac3VuLmNvbT6JARUDBRA0Z5icfpHfi79/fM0BARwaB/sG
        YHj+fpvMgRZev/i0DyZX+s6YyMZKeJ4pVHeboFP7KaP0R+VvAP0qojK+6ITUyX2w
        R3eqeJPMbWqmOA/EAYkYE/xcqrq2ddSq2SG43530/TTOfY+ENXttltVhBdJ79KLx
        8fR2f9jLKJqQBu2MRKpy5EdJ1qmthKQm/SGTKRz8uncs5BtmJxkAbskuSi6Ys24E
        Pv0r97dW/uTfh7VM8+SA/hkCF6QVE1UzvgpKwEpoh2DZiuzvwAFqV/tINZRHGhCg
        TNLvyz+5yYXSAY3nr8UPzNJ9QUXrsmzBGDSlpqp3GO7kL0VHN//B/5GLSVcofzpA
        xj/JP+41N4sDJGkyCWwqiQEeBBABAgAJBQI+d0xwAhkBAAoJEPCJEJL9ultFpMgH
        9AzI8pmuPKxv3dQcuqZ+rJRsy2YYuuSkWpj97n5PFWvBGTSAu2+2wo3uLn8A596w
        n4MVShtx5SC2rMKKZABJ8ObqtbbS1tQaIJmPg471qmnHjazeqbPfPwpQHzQ66cje
        De/3QbxBD/rPXV2SiyECed0qRsbuC9Oo3TonrJBOp6+Hs6jSkjGvQeJjvutuklMN
        A9TOd0CKN1RiEUWl4zwef7cmHWjWyfC64l8pqMFLC7XrYE7pXAL2Y6pi8Ta5njGL
        1dWryWzSDMCEunOt5wiuUYqZ+BXvy11kp2iKmi56ioTg5UHxGJqr6oZONDwMDIhW
        sI9v1kuHhJuWz8DZiZO1i7QgR2FyeSBFbGxpc29uIDxnZmVAaW50ZXJoYWNrLm5l
        dD6JARQDBRA+d1WR8IkQkv26W0UBAXgsB/UROD8wayj9v7gMK3K9Idxk/3K16myl
        m0Q5mzFkXoLZ6EJ3wZlpxteR9oeTo2F/5tJ0k9SFNaeIfFuipVGz9y+iDHHVKyQw
        kDGg7YB5+fK1siebpUnIemvhmngrUzLnmbOJDpBy+UukRGjRLhDsuEXN8fpGb27d
        ddo2odK31nR9OpRPGo/F2mkduatD28MMPVn4RpOKw8Nx7PIIxVPnTXGgfLY2PDOO
        Dk5he7KszA3rJul9Dof0Ii9nLHlOXiHwXWFx7le66vwlHCIaNwpvU8BXSeIgbKDA
        ZzFMfUHsKyTdMo9l+ByDk/jLsGsvZ61tROShVWSwO0rC8pKa3sVmSMy0C2dmZUBz
        dW4uY29tiQETAwUQP3plwvCJEJL9ultFAQGRDgfwmhqrrlACqYAr2a2yFoex0gIz
        NrTQvMjRWw5EyzoGu9KMQ5ilsBIpIHCcA6LY/Y6rb0qsrP7Pu0Z082uuQAlfpRzs
        i4lHsZDOeKKAiw7G3bJO+fDpkwYPHC7YFObof45Y71BWO+OBfKrMb73ZfgYYGKIc
        tECofkVO3fvNHNEeDIEzhvY2o783JOGbdN34P5NcLre69eLPF3KNhonLQMVxlNmh
        0kwl5rUckRPAPy4WgKv/VQEZtXSPmx9t4x3jUjc+yDtSdvTnBMwEHUU3/Pn8TICa
        XsvFX/55u0POntxFoi1A+0UpsCGrGpdzv1q7tRmFsF5aOP1Um79Qg1O/5060Gkdh
        cnkgRWxsaXNvbiA8Z2ZlQHN1bi5jb20+iQEUAwUQP3pmAvCJEJL9ultFAQF1twf0
        CAY7B8Nb74w+mYYyHS+UXCrPQR21vs5DjzuKooX7j6pJHDQqhfss24NLBvvpufZa
        uTE27fDIx+HC0SK5cjGUTqoX/4nkMe+HM87vPcChbS3lTGT+yxVjyiQ9BIei5mX2
        QTl9RkS3ZDXNux32uONDRX7dykNX6fYkKRGserWHhdXlHppmmvLodKCK/sZkkqzf
        VT4r9ytfpXBluelOV93X8RUz4ecZcDm9e+IEG+pQjnvgrSgac1NrW5K/CJEOUUjh
        oGTrym0Ziutezhrw/gOeLVtkywsMgDr77gWZxRvw01w1ogtUdTceuRBIDANj+KVZ
        vLKlTCaGAUNIjkiDDgti
        =OuKj
      </wsse:BinarySecurityToken>

      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <!-- in general include a ds:Reference for each wsa: header
               added according to SOAP binding -->

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

          <!-- include the To in the signature -->
          <ds:Reference URI="#to">...</ds:Reference>

          <!-- include the Action in the signature -->
          <ds:Reference URI="#action">...</ds:Reference>

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

          <!-- bind security token -->
          <ds:Reference URI="#bst">...</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>

    </wsse:Security>
  </s:Header>
  <s:Body wsu:Id="MsgBody">
```

```
    <!-- payload -->
  </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.

6.6.1. Identity Token Requirements

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 including the following: ~~token.~~

- A SAML 2.0 assertion element (`saml2:Assertion`) ~~may be used~~ as profiled~~an identity token. This usage is defined~~ in the Security Mechanisms SAML profile [LibertySecMech20SAML]. This ~~A WSS Binary Security Token may also be used as an identity token,~~ is a `saml2:Assertion`,~~it~~ and not a `saml2:EncryptedAssertion`, `saml2:NameID`, ~~attribute~~ or `saml2:EncryptedID`. ~~definition.~~
- An ~~A WSS SecurityTokenReference element may also be used~~ opaque~~to reference~~ value, for example~~token. Other~~ a `saml2:EncryptedAssertion`, `saml2:NameID`,~~XML~~ or `saml2:EncryptedID`, WSS Binary~~definitions~~ Security Token, or non-SAML~~be~~ values.

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 [LibertySecMech20SAML].

A WSS SecurityTokenReference element may also be used to reference an identity token.

6.6.2. Token Policy

The token policy describes the nature of the identity token to be returned upon an identity token request, generally focusing on the nature of the identifier. Details are defined in [LibertyAuthn].

The `<TokenPolicy>` element is of complex type **TokenPolicyType**, and contains the following attributes and elements:

- **`validUntil`** [Optional]

  Indicates the duration for which the requestor~~token is expected~~ would like the token to be valid. The responder MAY disregard the value in favor of its own policies.

- **`issueTo`** [Optional]

  Identifies the party to whom the identity token should be issued, if not otherwise apparent from the request or policy content. Note that this is usually not the party requesting the token, but generally a WSP the requester wishes to access.

  For example, a `samlp:NameIDPolicy` element may be included in the `TokenPolicy` element, and, in some cases, the value of the associated `SPNameQualifier` attribute will already indicate the party to whom the token is being issued, making use of `issueTo` unnecessary.

- **`type`** [Optional]

  Specifies the type of identity token to be returned upon an identity token~~to~~ request. If no type is specified, then~~which~~ the type of token returned is Opaque and need not necessarily~~this~~ be understood by the requestor.

  The value of the type attribute is a URI. The following are defined in this document:

  - SecMech-SAML-2.0-Assertion:

    - This MUST be a SAML 2.0 assertion (`saml2:Assertion`) as profiled in the Security Mechanisms SAML Profile. This is a~~token~~ `saml2:Assertion`, and~~as~~ not a `saml:EncryptedAssertion`, `saml:NameID`, or `saml:EncryptedID`, which are all considered Opaque types.

    - A~~outlined~~ `samlp2:NameIDPolicy` element SHOULD be included in the TokenPolicy element.

    - URI value: urn:liberty:security:2006-08:IdentityTokenType:SAML20Assertion

  - ~~Security~~

    Opaque:

    - The format is not specified and may be any format chosen by the IdP including, but not limited to,~~Mechanisms~~ SAML assertions, Encrypted Assertions, NameIDs, Encrypted NameIDs, WSS Binary Security Tokens, or other forms. ~~profile.~~

    - URI value: urn:liberty:security:2006-08::IdentityTokenType:Opaque

- **`wantDSEPR`** [Optional]

  Specifies whether the requestor would like the~~is~~ token to include a WSF 2.0 Endpoint Reference for the Discovery Service in a token returned by that Discovery Service. The default value is 'true'.

- **`Any Attribute`** [Zero or More]

  Any attribute can be used to describe other characteristics of the desired identity token. The wildcard is necessary because of the arbitrary nature of identity tokens.

- **Any Element** [Zero or More]

  Any element can be used to describe other characteristics of the desired identity token. The wildcard is necessary because of the arbitrary nature of identity tokens.

  ~~In the specific case of SAML-flavored identity tokens, a <samlp2:NameIDPolicy> element SHOULD be used.~~

```
<xs:complexType name="TokenPolicyType">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="validUntil" type="xs:dateTime" use="optional"/>
  <xs:attribute name="issueTo" type="xs:anyURI" use="optional"/>
  <xs:attribute name="type" type="xs:anyURI" use="optional"/>
  <xs:attribute name="wantDSEPR" type="xs:boolean" use="optional" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>

<xs:element name="TokenPolicy" type="sec:TokenPolicyType"/>
```

Figure 1. Element <TokenPolicy> Schema Fragment

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 that 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 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~~ sends a request to the~~onto~~ next service provider. This may be done by forwarding~~until~~ the ~~destination~~ request it received, acting~~reached. This~~ as~~mechanism allows~~ a proxy, or by generating~~performed,~~ a new request.~~provider proxies~~ This may be done until~~to~~ the destination service provider is reached.

An example is a browser client accessing a portal that 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.

Service providers may rely upon a security token passed with each request to make an authorization decision based on authentication, authorization and possibly other information contained within the token. The security token is unique to the service provider that consumes it, for example the principal ultimately invoking the destination service (the assertion subject) is conveyed using a name identifier appropriate to the service provider.

Note that the service provider itself may act as a policy decision point, or may use some other system entity as a policy decision point. How authorization is implemented is outside the scope of this specification, apart from the information conveyed in the message to enable such decisions.

The security token is passed in the `<wsse:Security>` header in the SOAP header block, as part of the SOAP request to a service provider. It is obtained by the service requestor as part of the discovery operation used to determine the endpoint information for the web service provider to whom the request is sent. When the Discovery Service returns a WS-Addressing endpoint reference (EPR) as profiled in the Discovery Service specification, it includes a security assertion appropriate for the requestor to transmit to the web service provider. This assertion is signed by the assertion issuer, e.g., the Discovery Service.

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

The `<TransitedProviderPath>` 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 `<TransitedProviderPath>`. To be meaningful in making 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 `<TransitedProviderPath>` since it is known to the message recipient as the sender of the message. The identity of this last transited provider MUST be recorded in the assertion, however, for example as part of the SAML assertion confirmation method.

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 7. Transited Providers

Party:	**A**	**B**	**C**	**D**	**E**	**F**
Assertion Contains:						
subject = principal = invoker		A(v)	A(w)	A(x)	A(y)	A(z)
sender(assertion confirmation method)			B	C	D	E
Provider Chain				(B)	(B,C)	(B,C,D)

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.

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 a bootstrap EPR in the security token it received with the request. This EPR indicates how to reach a Discovery Service for finding the next Web Service Provider, and this EPR includes a security token appropriate for the WSP to use in making a request to the DS. The DS may have included the <TransitedProviderPath> element in the security~~this~~ token contained in the bootstrap EPR, or may have included other information useful to the DS to perform the next step. Information that the DS may include is out of scope of this specification.

The WSP then sends a query to this Discovery Service using the bootstrap security token it received, placing it in the <wsse:Security> header block (and providing confirmation as necessary). Upon receipt the Discovery Service may use this security token in conjunction with the identity of the WSP indicated by the token to create a <TransitedProviderPath> (if needed) to place in the security token provided with the EPR for the next WSP.

When the Discovery Service creates the security token, it will map the name identifier of the assertion subject to a name identifier 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 token from the Discovery Service as part of the EPR, 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 <TransitedProviderPath>. Each token is a new token, with updated Subject name identifier and path information and 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 <TransitedProviderPath> is used, a single <TransitedProviderPath> element MUST be used to contain the information about all of the transited WSPs, in a single element. (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 <TransitedProviderPath> is included in a token, it contains <ProviderID> elements to indicate the identity of each transited WSP to the recipient. 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 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. TransitedProviderPath Schema

The <TransitedProviderPath> is used to identify the WSPs that are transited, apart from the last WSP that is transited. The intended usage of this element is to provide the authorization decision point associated with the final service provider transited WSP path information necessary to make an authorization decision.

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

```

<xs:complexType name="TransitedProviderPathType">
```

```
  <xs:sequence>
      <xs:element ref="sec:TransitedProvider" minOccurs="1"
                  maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:element name="TransitedProviderPath" type="sec:TransitedProviderPathType"/>


```

Note that a Discovery Service may decide to carry state information elsewhere in the assertion, for example in the Advice element of the SAML assertion. How this is done is outside the scope of this specification.

7.3.1.2. TransitedProvider Schema

A Discovery Service uses the `<TransitedProvider>` element to supply information about a single transited provider.

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

```

<xs:complexType name="TransitedProviderType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="timeStamp" type="xs:dateTime"
                    use="optional" />
      <xs:attribute name="confirmationURI" type="xs:anyURI"
                    use="optional" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="TransitedProvider" type="sec:TransitedProviderType" />


```

The semantics around the `<TransitedProvider>` element is as follows:

- The URI value of the `<TransitedProvider>` element is a URI determined by the Discovery Service. Typically it will be a ProviderID as defined in the Discovery Service specification.
- The OPTIONAL timestamp attribute is the time the message transited the provider. This is an approximate value since clock synchronization should not be expected to be accurate.
- The confirmationURI indicates the confirmation method used by the transited provider to confirm its identity to the Discovery service when obtaining the EPR to send the request to the next WSP.

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 and Integrity (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 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 profile.

7.5.1. Processing Rules

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

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

8. Schema

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:liberty:security:2006-08"
      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
      xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:sec="urn:liberty:security:2006-08"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      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"
      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="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#"
      schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
    <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-util-1.0.xsd"/>

<xs:complexType name="TokenPolicyType">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="validUntil" type="xs:dateTime" use="optional"/>
  <xs:attribute name="issueTo" type="xs:anyURI" use="optional"/>
  <xs:attribute name="type" type="xs:anyURI" use="optional"/>
  <xs:attribute name="wantDSEPR" type="xs:boolean" use="optional" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>

<xs:element name="TokenPolicy" type="sec:TokenPolicyType"/>

<xs:complexType name="TransitedProviderType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="timeStamp" type="xs:dateTime"
                   use="optional" />
      <xs:attribute name="confirmationURI" type="xs:anyURI"
                   use="optional" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="TransitedProvider" type="sec:TransitedProviderType" />

<xs:complexType name="TransitedProviderPathType">
  <xs:sequence>
      <xs:element ref="sec:TransitedProvider" minOccurs="1"
                  maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:element name="TransitedProviderPath" type="sec:TransitedProviderPathType"/>
<!--
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:attribute name="usage" type="xs:anyURI" use="optional" />
</xs:complexType>

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

</xs:schema>
```

References

Normative

[LibertySecMech11] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.1, Liberty Alliance Project (18 April 2004). *http://www.projectliberty.org/specs/*

[LibertySecMech12] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.2, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs/*

[LibertyAuthn] Hodges, Jeff, Aarts, Robert, Madsen, Paul, Cantor, Scott, eds. " Liberty ID-WSF Authentication, Single Sign-On, and Identity Mapping Services Specification ," Version 2.0-errata-v1.0, Liberty Alliance Project (28 November, 2006). *http://www.projectliberty.org/specs*

[LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 2.0-01, Liberty Alliance Project (21 November 2004). *http://www.projectliberty.org/specs*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty ID-WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007). *http://www.projectliberty.org/specs*

[LibertyDisco] Cahill, Conor, Hodges, Jeff, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (29 November, 2006). *http://www.projectliberty.org/specs*

[LibertyIDWSFv20Errata] Champagne, Darryl, Lockhart, Rob, Tiffany, Eric, eds. "Liberty ID-WSF 2.0 Errata," Version 1.0, Liberty Alliance Project (13 April, 2007). *http://www.projectliberty.org/specs*

[LibertyGlossary] Hodges, Jeff, eds. "Liberty Technical Glossary," Version v2.0, Liberty Alliance Project (30 July, 2006). *http://www.projectliberty.org/specs*

[SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (2 September 2003). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1," SAML v1.1, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf*

[SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf*

[SAMLBind11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (2 September 2003). "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1," SAML V1.1, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/download.php/3405/oasis-sstc-saml-bindings-1.1.pdf*

[SAMLBind2] Cantor, Scott, Hirsch, Frederick, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf*

[LibertySecMech20SAML] Hirsch, Frederick, eds. "ID-WSF 2.0 SecMech SAML Profile," Version 2.0-errata-v1.0, Liberty Alliance Project (08 November, 2006). *http://www.projectliberty.org/specs*

[wss-sms11] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (June 28, 2005). "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)," Public Review Draft - 28 June 2005, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/download.php/13397/wss-v1.1-spec-pr-SOAPMessageSecurity-01.pdf*

[wss-saml] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (December 1, 2004). Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf* "Web Services Security: SAML Token Profile," OASIS Standard V1.0 [OASIS 200412],

[wss-saml11] Monzillo, Ronald, Kaler, Chris, Nadalin, Anthony, Hallam-Baker, Phillip, eds. (June 28, 2005). Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/download.php/13405/wss-v1.1-spec-pr-SAMLTokenProfile-01.pdf* "Web Services Security: SAML Token Profile 1.1," OASIS Public Review Draft 01,

[wss-x509] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (March, 2004). Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf* "Web Services Security: X509 Certificate Token Profile," OASIS Standard V1.0 [OASIS 200401],

[wss-kerb] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (May 5, 2003). Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/download.php/1049/WSS-Kerberos-03.pdf* "Web Services Security: Kerberos Token Profile," Draft WSS-Kerberos-03,

[XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and Processing," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmldsig-core*

[xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (10 December 2002). "XML Encryption Syntax and Processing," W3C Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlenc-core/*

[RFC3268] Chown, P., eds. (June 2002). "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)," RFC 3268., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3268.txt*

[SOAPv1.2] "SOAP Version 1.2 Part 1: Messaging Framework," Gudgin, Martin, Hadley, Marc, Mendelsohn, Noah, Moreau, Jean-Jacques, Nielsen, Henrik Frystyk, eds. World Wide Web Consortium W3C Recommendation (07 May 2003). *http://www.w3.org/TR/2003/PR-soap12-part1-20030507/*

[SSL] Frier, A., Karlton, P., Kocher, P., eds. (November 1996). Netscape Communications Corporation "The SSL 3.0 Protocol," *http://www.netscape.com/eng/ssl3/*

[RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engineering Task Force (March 1997). *http://www.ietf.org/rfc/rfc2119.txt*

[RFC4346] Dierks, T., Rescorla, E., eds. (April 2006). "The Transport Layer Security (TLS) Protocol," Version 1.1 RFC 4346, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc4346.txt*

[Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlschema-1/*

[WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds. World Wide Web Consortium W3C Recommendation (9 May 2006). *http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/*