Liberty ID-WSF Security Mechanisms

Version: v2.0-03

Editors:
Gary Ellison, Sun Microsystems, Inc.
Paul Madsen, Entrust, Inc.

Contributors:
Robert Aarts, Nokia Corporation
Carolina Canales-Valenzuela, Ericsson
Conor Cahill, AOL Time Warner, Inc.
Scott Cantor, Internet2, The Ohio State University
Frederick Hirsch, Nokia Corporation
Jeff Hodges, Sun Microsystems, Inc.
John Kemp, Nokia Corporation
John Linn, RSA Security Inc.
Jonathan Sergent, Sun Microsystems, Inc.
Greg Whitehead, Trustgenix, Inc.

Abstract:

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

Filename: draft-liberty-idwsf-security-mechanisms-v2.0-03.pdf

Notice

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

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

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

Revision History
Revision: 2.0-03 **Date:** 22 Novermber 2004
Version, and schema update
Revision: 2.0-02 **Date:** 9 Novermber 2004
Editorial
Revision: 2.0-01 **Date:** 4 October 2004
Overhaul for Samlv2.0

Contents

1. Abstract

This document specifies security protocol mechanisms for securing the consumption of identity-based web services. An identity-based web service is a particular type of a web service that acts upon some resource to either retrieve information about an identity, update information about an identity, or to perform some action for the benefit of some identity. This document describes authentication mechanisms which are factored into the authorization decisions enforced by a given identity-based web service. The specified mechanisms provide for authentication, signing and encryption operations. XML-Signature ([XMLDsig]) and XML-Encryption ([xmlenc-core]) are utilized to provide the associated transformations and processing semantics to accommodate the message authentication and protection functionality. OASIS Web Services Security SOAP Message Security ([wss-sms]) compliant header elements communicate the relevant security information, i.e., a SAML [SAMLCore11] or [SAMLCore2] assertion, along with the protected message.

2. Overview of Identity-Based Web Services Authorization (Informative)

This section provides a perspective of some of the authorization obligations an identity-based web service may assume.

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.

Identity-based web services may be accessed by system entities. The access may be direct or with the assistance of an intermediary. To access an identity-based web service a system entity must interact with a specific service instance that exposes some resource.

Given the above description, we strongly believe that access control policies must be enforced by identity-based web services. The authorization decision to access an identity-based web service instance offering a specific resource may be made locally (that is at the entity hosting the resource) or remotely. Regardless of whether the policy decision point (PDP) is distributed or not a policy enforcement point (PEP) will likely be implemented by the entity hosting or exposing the resource.

In most cases, the service requester directly interacts with the identity-based web service, thus the identity-based web service may implement both the PEP and the PDP. Under these circumstances the authorization decision, at a minimum, should be based on the authenticated identity of the service requester and the resource for which access is being requested.

However, an identity-based web service may rely upon a trusted third party (TTP) to make coarse policy decisions. It is also likely that the TTP will act as a Policy Information Point (PIP) such that it can convey information regarding the resource and the policy it maintains. This scenario might be deployed in the event that the principal is unable to actively authenticate to the identity-based web service. One such scenario is where a TTP provides a bridge function to introduce new participants to the identity service. The result of any such policy decision made by the TTP must be presented to the entity hosting the identity-based web service. Of course this does not preclude the identity-based web service from making additional policy decisions based on other criteria.

Our definition of an identity-based web service mentioned the notion of the service performing an action for the benefit of an identity. To fully appreciate the possibilities this notion suggests one must recognize scenarios whereby peer entities may need to represent or perform actions on behalf of other system entities. It may also be the case that the identity-based web service must consider the status of the resource owner for a given request to access a resource.

To support the case where an intermediary accesses a resource on behalf of another system entity, the identity-based web service may rely upon a TTP to make policy decisions and issue statements that allow the service requester to act on behalf of a different system entity.

3. Notation and Terminology

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

3.1. Notational Conventions

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

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

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

3.2. Namespace

The following namespaces are referred to in this document:

Table 1. Namespaces

Prefix	Namespace
sec	*urn:liberty:sec:2004-12*
sb	*urn:liberty:sb:2003-08*
disco	*urn:liberty:disco:2003-08*
saml	*urn:oasis:names:tc:SAML:2.0:assertion*
S	http://www.w3.org/2002/12/soap-envelope
ds	http://www.w3.org/2000/09/xmldsig#
xenc	http://www.w3.org/2001/04/xmlenc#
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

This specification uses the following typographical conventions in text: <Element>, <ns:ForeignElement>, Attribute, Datatype, OtherCode.

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

3.3. Terminology

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

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

- Recipient – entity which receives a message that is the ultimate processor of the message
- Sender – the initial SOAP sender. A sender is a proxy when its identity differs from the invocation identity.

- Proxy – entity whose authenticated identity, according to the recipient, differs from that of the entity making the invocation.
- Trusted Authority – a Trusted Third Party (TTP) that issues, and vouches for, SAML assertions
- Invocation Identity – party invoking a service.
- Service – invocation responder, providing a service. Ultimate message processor.

4. Security Requirements (Informative)

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

4.1. Security Requirements Overview

There are multiple facets this security specification considers:

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

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

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

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

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

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

4.2. Common Requirements

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

- Messages may need to be kept confidential and inhibit unauthorized disclosure, either when in transit or when stored persistently. Confidentiality may apply to the entire message, selected headers, payload, or XML portions depending on application requirements.
- Messages 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.
- Message responses must correspond to message requests and attempts to replay requests or responses should be detected. Likewise the attempt to substitute requests or responses should be detected. Transaction integrity requires that messages be timely and related to each other.
- The privacy requirements of the participants with respect to how their information is shared or correlated must be ensured.

4.3. Peer Authentication Requirements

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

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

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

4.4. Message Correlation Requirements

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

4.5. Privacy Requirements

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

4.6. Service Availability Requirements

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

4.7. Resource Access Authorization Requirements

Previously we mentioned the notion of conveying both a *sender identity* and an *invocation identity*. In doing so the framework accommodates a restricted proxy capability whereby a consumer 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 distribute an assertion authorizing the intermediary 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 the proxy to access a web service, the recipient should still implement a policy enforcement point.

5. Confidentiality and Privacy Mechanisms

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

5.1. Transport Layer Channel Protection

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

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

  - TLS_RSA_WITH_RC4_128_SHA
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

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

  - TLS_RSA_WITH_AES_CBC_SHA
  - TLS_DHE_DSS_WITH_AES_CBC_SHA

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

- Other security protocols (e.g. Kerberos, IPSEC) MAY be used as long as they implement equivalent security measures.

5.2. Message Confidentiality Protection

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

Please note that this mechanism does not fully address the privacy and confidentiality requirements of information supplied by a trusted authority which is subsequently carried in the `<S:Header>` which is not to be revealed to the entity interacting with the recipient. For example the authorization data may contain sensitive information. To accommodate this requirement the trusted authority and ultimate recipient MUST rely upon the mechanisms specified in Encrypted Name Identifiers (Section 5.3.1) and in Encrypted Attributes (Section 5.3.2) SHOULD be used.

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 resource identifier, which has some association with the identity-based web service and the principal on whose behalf the sender is acting.

5.3.1. Encrypted Name Identifiers

The identity 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. To accomplish this securely the `<saml:Subject>` MUST contain a `<saml:EncryptedID>` following the processing rules and recommendations specified in [SAMLCore2].

5.3.2. Encrypted Attributes

At times it may be necessary to privacy protect the contents of a resource identifier (see [LibertyDisco]), which is expressed in the form of a URI to deter the release of sensitive information to an intermediary. The [SAMLCore2] specification defines an encrypted form of an attribute statement with the `<saml:EncryptedAttribute>` schema element. This specification relies upon the semantics defined in [SAMLCore2] to fulfill this privacy requirement. Thus the processing rules defined by [SAMLCore2] for the `<saml:EncryptedAttribute>` element MUST be followed.

6. Authentication Mechanisms

This specification defines a set of authentication mechanisms, labeled by URIs, and the security properties they engender. The multiplicity of mechanisms specified is necessary to accommodate various deployment scenarios. Each identifier represents two security properties for a given mechanism:

- Peer Entity Authentication
- Message Authentication

For either of the properties a value of "null" indicates that the particular security property is not supported by the mechanism. For the peer entity authentication property, the qualifier indirectly indicates which actor(s) is authenticated in a given interaction. For the message authentication property the qualifier describes the security profile utilized to secure the message.

The following table summarizes all the authentication mechanism identifiers defined as of the publication of this specification. Not all of these mechanisms and there semantics are defined in this version of the specification. Specifically, [SAMLCore11] based identifiers are defined in a previous version of this specification [LibertySecMech].

Each URI is of the form *urn:liberty:security:yyyy-mm:peer mechanism:message mechanism*.

Table 2. Authentication Mechanisms

URI	Peer Entity	Message	Normative Spec.
urn:liberty:security:2003-08:null:null	No	No	[LibertySecMechV20]
urn:liberty:security:2003-08:null:X509	No	Yes	[LibertySecMechV20]
urn:liberty:security:2003-08:null:SAML	No	Yes	[LibertySecMech]
urn:liberty:security:2004-12:null:SAMLV2	No	Yes	[LibertySecMechV20]
urn:liberty:security:2004-04:null:Bearer	No	No	[LibertySecMechV20]
urn:liberty:security:2003-08:TLS:null	Recipient	No	[LibertySecMechV20]
urn:liberty:security:2003-08:TLS:X509	Recipient	Yes	[LibertySecMechV20]
urn:liberty:security:2003-08:TLS:SAML	Recipient	Yes	[LibertySecMech]
urn:liberty:security:2004-12:TLS:SAMLV2	Recipient	Yes	[LibertySecMechV20]
urn:liberty:security:2004-04:TLS:Bearer	Recipient	No	[LibertySecMechV20]
urn:liberty:security:2003-08:ClientTLS:null	Mutual	No	[LibertySecMechV20]
urn:liberty:security:2003-08:ClientTLS:X509	Mutual	Yes	[LibertySecMechV20]
urn:liberty:security:2003-08:ClientTLS:SAML	Mutual	Yes	[LibertySecMech]
urn:liberty:security:2004-12:ClientTLS:SAMLV2	Mutual	Yes	[LibertySecMechV20]
urn:liberty:security:2004-04:ClientTLS:Bearer	Mutual	No	[LibertySecMechV20]

6.1. Authentication Mechanism Overview (Informative)

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

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

The peer entity authentication mechanisms defined by this specification leverage the authentication features supplied by SSL 3.0 [SSL] or TLS 1.0 [RFC2246]. The mechanism identifier describes whether the recipient ("TLS") is unilaterally authenticated or whether each communicating peer ("ClientTLS") is mutually authenticated to the other

peer. The peer entity authentication mechanisms (Section 6.2) are best suited for direct message exchanges between end systems and when the message exchange may be sufficiently trusted to not require additional attestation of the message payload. However this does not obviate the processing of subject confirmation obligations but rather enables alternative and potentially optimized processing rules. Such optimizations are a matter of security policy as it applies to the trust model in place between communicating entities.

The message authentication mechanisms indicate which attestation profile is utilized to ensure the authenticity of a message. These message authentication facilities aid the deployer in the presence of intermediaries. The different message authentication mechanims are suited (but not necessarily restricted) to different authorization models:

- The X.509 v3 Certificate mechanism (Section 6.3.1) 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 (Section 6.3.2) 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 6.3.3) is based on the presence of a *bearer token* in the security header of a message for which the sender does not explicitly demonstrate the right to lay claim to. In this case, the bearer token is verified for authenticity rather than proving the authenticity of the message.

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

Not all peer entity authentication and message authentication combinations make sense in a given setting. Again this is a matter of security policy and the trust model the policy accords. For example, in a conventional setting where peer entity authentication is relied upon to ensure the authenticity, confidentiality and integrity of the transport in conjunction with message authentication to assure message authorship, intent and retention of the act of attestation then the mechanism *urn:liberty:security:2003-08: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:2003-08:ClientTLS:X509* seems equivalent to *urn:liberty:security:2003-08:ClientTLS:null* in such a setting. A similar argument can be made for the *urn:liberty:security:2004-12:ClientTLS:SAMLV2* mechanism. The relationship between the identity authenticated as a result of peer entity authentication and the identity authenticated (or implied) from message authentication may diverge and describe two distinct system entities for example, a system principal and a user principal respectively. The identities may also be required to reflect the same system entities. This is a matter of deployment and operational policy and is out of scope for this specification.

6.2. Peer Entity Authentication

The Peer entity authentication mechanisms prescribed by this specification all rely upon the inherent security properties of the TLS/SSL protocol (sometimes referred to as transport-level security); the different mechanims differentiated by how the message is authenticated. The mechanisms described below have distinct security properties regarding which peers in a message exchange are authenticated. 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 the following URIs are described in a prior version of this specification [LibertySecMech]:

- *urn:liberty:security:2003-08:TLS:SAML*

The semantics and processing rules for the following URIs are described in this specification. These URIs support unilateral (recipient) peer entity authentication:

- *urn:liberty:security:2003-08:TLS:null*
- *urn:liberty:security:2003-08:TLS:X509*
- *urn:liberty:security:2004-12:TLS:SAMLV2*
- *urn:liberty:security:2004-04:TLS:Bearer*

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

The latter two mechanisms MAY be used in conjunction with message authentication mechanisms defined by this specification.

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.

6.2.2. Mutual Peer Entity Authentication

The semantics and processing rules for the following URIs are described in a prior version of this specification [LibertySecMech]:

- *urn:liberty:security:2003-08:ClientTLS:SAML*

The semantics and processing rules for the following URIs are described in this specification. These URIs support mutual (sender and recipient) peer entity authentication:

- *urn:liberty:security:2003-08:ClientTLS:null*
- *urn:liberty:security:2003-08:ClientTLS:X509*
- *urn:liberty:security:2004-12:ClientTLS:SAMLV2*
- *urn:liberty:security:2004-04:ClientTLS:Bearer*

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.

The latter two URIs indicate that the mechanism may be used in conjunction with message authentication mechanisms defined by this specification.

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.

6.3. Message Authentication

The non-null message authentication mechanisms prescribed by this specification generally rely upon the integrity properties imbued by the application and verification of digital signatures over elements of the message header and payload. The mechanisms described below 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.

6.3.1. X.509 v3 Certificate Message Authentication

The following URIs define X509 based unilateral (sender) message authentication mechanisms:

- *urn:liberty:security:2003-08:null:X509*
- *urn:liberty:security:2003-08:TLS:X509*
- *urn:liberty:security:2003-08:ClientTLS:X509*

These mechanisms utilize 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 a subject confirmation key associated with the X.509 certificate. 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.

6.3.1.1. Sender Processing Rules

- The construction and insertion of the `<wsse:Security>` element MUST adhere to the rules specified in the [wss-sms] and [wss-x509].

- The sender MUST demonstrate possession of a subject confirmation key.
  For deployment settings which REQUIRE independent message authentication, the obligation MUST be accomplished by signing elements of the message and decorating the `<wsse:Security>` element with the signature.
  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 the *urn:liberty:security:2003-08:ClientTLS:X509* mechanism.

- If peer entity authentication is not in use and the message is bound with [LibertySOAPBinding] the sender MUST sign:

  - The `<sb:Correlation>` header block element.

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

  - All sub-elements of the `<S:Body>`.

- If the message is signed then 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 which SHOULD carry a `<wsse:SecurityTokenReference>` element.

6.3.1.2. Recipient Processing Rules

- The recipient MUST locate the `<wsse:Security>` element for which it is the target. This MUST adhere to the syntax and processing rules specified in [wss-sms] and [wss-x509].

- If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the recipient MUST locate the `<ds:KeyInfo>` element bearing a security token. This token MUST be unambiguously verified to be referring to the same certificate and key used in establishing peer entity authentication.

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

- If peer entity authentication is not in use and the message is bound with [LibertySOAPBinding] the recipient MUST verify the signature covers the following elements:

  - The `<sb:Correlation>` header block element.

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

  - All sub-elements of the `<S:Body>`.

6.3.2. SAML Assertion Message Authentication

The semantics and processing rules for the following URIs are described in a prior version of this specification [LibertySecMech]:

- *urn:liberty:security:2003-08:null:SAML*
- *urn:liberty:security:2003-08:TLS:SAML*
- *urn:liberty:security:2003-08:ClientTLS:SAML*

The semantics and processing rules for the following URIs are described in this specification. These URIs indicate unilateral SAML-based message authentication mechanisms:

- *urn:liberty:security:2004-12:null:SAMLV2*
- *urn:liberty:security:2004-12:TLS:SAMLV2*
- *urn:liberty:security:2004-12:ClientTLS:SAMLV2*

These mechanisms utilize the Web Services Security SAML Token Profile [wss-saml] as the means by which the message sender authenticates to the recipient. In general these mechanisms assume that a TTP issues an assertion which includes an `<saml:AuthnStatement>` and other statements which apply to the entity identified within the `<saml:Subject>` element. The `<saml:AuthnStatement>` describes the authentication event of the subject to the issuing authority. For this and any other statements in the assertion to be considered trustworthy, the subject confirmation obligations specified in the `<saml:Subject>` element must be met by the sender.

As a security precaution, the issuer of the assertion MUST include a `<saml:AudienceRestriction>` element that specifies the intended consumer(s) of the assertion. One `<saml:Audience>` element MUST be set to contain the unique identifier of the intended recipient, as described by the name identifier Format URI of *urn:oasis:names:tc:SAML:2.0:nameid-format:entity* as specified in [SAMLCore2]. The recipient MUST validate that it is the intended consumer before relying upon the assertion. The assertion MAY contain additional `<saml:Audience>` elements that specify other intended parties.

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 the mechanisms defined in Section 6.3.1 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.

For deployment settings which require message authentication independent of peer entity authentication, then the sending peer MUST perform message authentication by confirming in accordance with the obligations described by the `<saml:SubjectConfirmation>` element.

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.

6.3.2.1. Sender Processing Rules

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

- The sender MUST present the <saml:Assertion> (as security token) by inserting it as a child of the <wsse:Security> element.

- The sender MUST adhere to its subject confirmation obligation in accordance with the semantics of the confirmation method described by one of the <saml:SubjectConfirmation> elements carried within the <saml:Subject>.
  For deployment settings which REQUIRE independent message authentication, the obligation MUST be accomplished by signing elements of the message and decorating the <wsse:Security> element with the signature.
  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 subject confirmation element. To accommodate this, the assertion issuing authority MUST construct the assertion such that the confirmation key can be unambiguously verified to be the same certificate and key used in establishing peer entity authentication. This is necessary to mitigate the threat of a certificate substitution attack. It is RECOMMENDED that the certificate or certificate chain be bound to the subject confirmation key.

- If peer entity authentication is not used and the message is bound to SOAP with [LibertySOAPBinding] the sender MUST sign:

  - The <sb:Correlation> header block element.

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

  - All sub-elements of the <S:Body>.

- 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 SHOULD 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 3.3.2 of [wss-saml].

6.3.2.2. Recipient Processing Rules

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

- The recipient MUST locate the <saml:Assertion> (security token) and the recipient MUST determine that it trusts the authority which issued the <saml:Assertion>.
  The recipient MUST validate the issuer's signature over the <saml:Assertion>. The recipient SHOULD validate the trust semantics of the signing key, as appropriate to the risk of incorrect authentication.

- The recipient SHOULD verify that at least one of the confirmation obligations specified in the <saml:SubjectConfirmation> element has been met.

- If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the recipient MUST locate the <ds:KeyInfo> element within <saml:SubjectConfirmation> element. This key MUST be unambiguously verified to be referring to the same certificate and key used in establishing peer entity authentication.

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

- If peer entity authentication is not in use and the message is bound with [LibertySOAPBinding] the recipient MUST verify the signature covers the following elements:

  - The `<sb:Correlation>` header block element.
  - All other header block elements that require the aforementioned security properties in accordance with the security requirements prescribed in their respective specification.
  - All sub-elements of the `<S:Body>`.

6.3.3. Bearer Token Authentication

The following URIs indicate bearer mechanisms:

- *urn:liberty:security:2004-04:null:Bearer*
- *urn:liberty:security:2004-04:TLS:Bearer*
- *urn:liberty:security:2004-04:ClientTLS:Bearer*

These mechanisms rely upon bearer semantics as a means by which a message sender conveys to the recipient the senders identity. This specification only describes common markup and processing rules that MUST be adhered to. The actual semantics of the content and verification requirements of a bearer token are specific to the token type.

For example, a bearer token with a `wsse:ValueType` attribute of `http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-` [wss-saml] could contain statements describing other participants to a transaction. For such a scenario, it is presumed that the subject confirmation obligations described by the statements within the assertion would be of type, `urn:oasis:names:tc:SAML:1.0:cm:bearer` [SAMLBind2] and that the relying party would validate the assertion in accordance with the processing rules of [SAMLCore2]. Particular attention must be paid to the proper validation of the `<saml:AudienceRestriction>` element which specifies the intended consumer(s) of the assertion. In this case the assertion construction guidance in Section 6.3.2 would apply.

An example of a SAML bearer token can be found in Section 9.5.

This specification does not limit the types of bearer tokens which can be conveyed to the token forms profiled by [wss-sms], [wss-x509] or [wss-saml]. That is, custom tokens or tokens which are subsequently profiled after this specification is finalized could still leverage this mechanism providing the `wsse:ValueType` is understood by the producer and consumer of the token. See the example in Section 9.7.

These message authentication mechanisms only pertain to the bearer token within the message.

These mechanisms do not protect the integrity, authenticity or confidentiality of the bearer token and thus caution must be taken to not 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 inconjunction with or instead of an unprotected bearer mechanism.

6.3.3.1. Sender Processing Rules

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

- The sender SHOULD wrap the bearer token within a `<wsse:Embedded>` element and make it a child of a `<wsse:SecurityTokenReference>` as described in section 7.4 of [wss-sms].
  The sender SHOULD indicate the type of the token by specifying the `wsse:ValueType` attribute of the `<wsse:Embedded>` element.

- Alternatively the sender MAY simply insert the token within the `<wsse:Security>` header when the token type is well known or takes an obvious form (e.g. `<wsse:BinarySecurityToken>`.)

6.3.3.2. Recipient Processing Rules

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

- The recipient MUST locate the bearer token by processing `<wsse:Embedded>` elements within the `<wsse:SecurityTokenReference>` element.
  The recipient MUST process the token in accordance with the processing rules of the token type as indicated by the `wsse:ValueType` attribute of the `<wsse:Embedded>` element.

- Alternatively the recipient MAY be able to locate the token by it's well known schema type (e.g. `<wsse:BinarySecurityToken>`.)

7. Message Authorization Model

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 facilitate the exchange of resource access information to the relying party. As a PDP, the Trusted Third Party would adhere to the coarse access policies of the relying party insofar as ensuring which entities may attempt to access a given resource. 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.

7.2. Authorization Mechanism

The following mechanism description assumes that the Web Services Security SAML Token Profile [wss-saml] is utilized as the means by which the message sender authenticates to the message recipient. Each communicating peer performs message level authentication by fulfilling the subject confirmation obligation. Typically this is by demonstrating proof of possession of a subject confirmation key. The assertion issuer binds the subject confirmation key to the assertion by signing the assertion. This attestation provides assurance to the consumer of the assertion that the subject confirmation key is that of the intended sender. Thus the sender's subject confirmation key can be recognized by the recipient as belonging to the confirming peer. The assertion issuer should also bind a name identifier to the subject confirmation element. This name binding would serve as an aid in associating the application domain name of the sender with its confirmation key. Subsequent to the authentication of the sender the recipient can leverage this knowledge in support of the authorization model described below.

The authorization model supports the issuance of assertions that convey information regarding the resource to be accessed, the entity attempting to access the resource, the mechanism by which the accessing entity must use to confirm its identity to the recipient and the ability for the sending entity to access the resource on behalf of another system entity. This latter facility suggests the need to verify two distinct identities in a given message; the sender identity (the proxy) and the invocation identity (the subject). Thus the authorization model supports a constrained proxy mechanism that permits the confirming entity (a proxy) to access the resource on behalf of the asserted subject.

7.3. Authorization Data Generation

It is anticipated that a trusted service exists which aids in the discovery of identity-based web services. In support of this, a trusted authority [LibertyDisco] may issue an assertion, which is subsequently used in conjunction with the accessing of 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.

7.3.1. Processing Rules

The following processing rules describe the steps the assertion issuing authority takes to generate an assertion. It is out of scope for this specification to describe how assertions are requested and distributed. However it is presumed that in order for assertions to be generated that the requester has been authenticated and that the assertion issuing authority has enforced the necessary access controls to ensure that the assertions are released to authorized entities.

The assertion issuing authority constructs the assertion in accordance with the following rules:

- The assertion MUST indicate the invocation identity within the `<saml:Subject>` element of the assertion.
  The `<saml:Subject>` element MUST include at least one `<saml:SubjectConfirmation>` element. This element MUST have a `Method` attribute with a value of `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key`. The subject confirmation element MUST be specified with a `<saml:SubjectConfirmationData>` element qualified with an `xsi:type` of `saml:KeyInfoConfirmationDataType` as specified in [SAMLCore2].

- When the invocation identity represents the identity of the sender, the `<saml:Subject>` element is decorated as follows. Refer to Section 9.1.1 for an informative example.
  The name identifier element SHOULD include a `<saml:NameID>` element and the `Format` attribute value SHOULD be `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note: This identifier might assist the relying party in locating metadata concerning the subject of the assertion.
  The `<saml:SubjectConfirmation>` element SHOULD NOT be decorated with a `<saml:NameID>` element.

- When the invocation identity is NOT that of the sender (i.e., the sender is acting as a proxy on behalf of the subject) the `<saml:Subject>` element is decorated as follows:
  In an operational setting where the invocation identity (the subject) is only to be released to the relying party (the audience) then the name identifier element SHOULD be of type `<saml:EncryptedID>` and conform to the guidance in [SAMLCore2]. Refer to Section 9.1.2.2 for an informative example.
  In settings where the invocation identity does not call for privacy protections then the name identifier element SHOULD be conveyed using a `<saml:NameID>` element with a `Format` attribute which is appropriate for the operational setting. Refer to Section 9.1.2.1 for an informative example.
  To identify the confirming entity the `<saml:SubjectConfirmation>` element SHOULD contain a `<saml:NameID>` element with a `Format` attribute value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note: This identifier might assist the relying party in locating metadata concerning the confirming entity as well as help associate the name of the confirming entity in the application domain namespace with the key used for subject confirmation.

- The assertion issuing authority MAY describe the authentication status of the interacting party by including a `<saml:AuthnStatement>` element which MUST include a `<saml:AuthnContext>` element. Refer to Section 9.1.3 for an informative example.

- The assertion issuing authority MAY describe the resource for which sender intends to access at the relying party by including an `<saml:AttributeStatement>`.
  In an operational setting where the value of the attribute requires confidentiality protections then the attribute element SHOULD be of type `<saml:EncryptedAttribute>` and conform to the guidance in [SAMLCore2]. Refer to Section 9.1.4.2 for an informative example.
  If the confidentiality of the attribute is not a concern then the element SHOULD be conveyed using a `<saml:Attribue>`. Refer to Section 9.1.4.1 for an informative example.

- OPTIONALLY, the assertion issuer MAY include information that assists in building a chain of transited proxies. It is RECOMMENDED that the `<saml:Advice>` element be decorated with a `<saml:AssertionIDRef>` which is a reference to the assertion bearing it. Also as the chain builds the assertion should be augmented with a `<ProxyTransitedStatement>`. The issuer should include a `<saml:SubjectConfirmation>` for each proxy (except for the last) that has participated in the progression of assertion issuance. See Section 7.3.2 for a description of how the proxy chain is constructed.

- The assertion MUST be signed by the assertion issuing authority in accordance with the signing requirements specified in [SAMLCore2].

7.3.2. Proxy Chaining

Proxy chaining refers to scenarios in which a recipient, upon receiving a request from a sender, itself proxies the request onto the ultimate recipient (or some other intermediate proxy). In some operational settings it may be necessary to carry this chain of traversed proxies to the ultimate recipient. The following describes how the proxy chain is constructed through successive interactions between the involved proxies and the assertion issuer.

It is presumed that the assertion issuing authority decorates assertions with `<saml:AssertionIDRef>` within the `<saml:Advice>` element for assertions which it deems to be proxiable.

When a recipient receives a request for which it is necessary to proxy, it interacts with the assertion issuer and includes a `<ProxyTransitedStatement>` containing a `<SubjectConfirmation>` as its subject confirmation data. This claim SHOULD be in the form of a SAML assertion carried as a security token within the security header of the request to the assertion issuing authority.

The confirmation data sent to the assertion issuer includes the `<saml:AssertionIDRef>` of the assertion which the recipient received from the initial sender. The assertion issuer will use the `<saml:AssertionIDRef>` information to locate the initial sender's assertion and add it to the list of proxies transited.

The assertion issuer will create an `<saml:Assertion>` comprised of a `<ProxyTransitedStatement>` element which in turn contains `<saml:SubjectConfirmation>` elements for each of the proxies transited. Each `<saml:SubjectConfirmation>` element contains an instance of `<ProxyInfoConfirmationData>` as subject confirmation data.

It is recommended that this assertion be carried within an `<saml:Advice>` element of the assertion issued to the proxy.

See Section 9.4 for an example of a `<saml:Assertion>` carrying a `<ProxyTransitedStatement>` with multiple `<SubjectConfirmation>` elements.

7.4. Presenting Authorization Data

Interactions with identity-based web services may rely on the conveyance of authorization information. In general, the 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 Data Generation (Section 7.3) for details as to how this data is acquired and formulated.

7.4.1. Processing Rules

- The sender MUST authenticate to the recipient using one of the authentication mechanisms described in Message Authentication (Section 6.3).
  It is RECOMMENDED that the sender authenticate using the SAML Assertion Message Authentication and specifically conform to the processing rules specified in (Section 6.3.2.1).

7.5. Consuming Authorization Data

A recipient which 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).

The recipient determines the invocation identity by inspecting the `<saml:Subject>` element. If a proxy is involved in the communication then it's identity is carried within the `<saml:NameID>` element of the `<saml:SubjectConfirmation>` element in effect. Providing both the invocation identity and the proxy identity enables the recipient to tailor authorization policy to a finer degree of granularity. That is, the recipient generally uses the invocation identity to make its authorization decisions and potentially determine whether the proxy is permitted to access the resource on behalf of said invocation identity.

7.5.1. Processing Rules

- The recipient MUST authenticate the sender using one of the mechanisms described in Authentication Mechanisms (Section 6.3.2).
  It is RECOMMENDED that the sender authenticate using the SAML Assertion Message Authentication and specifically conform to the processing rules specified in (Section 6.3.2.2).

- The recipient MUST locate the `<saml:Assertion>` (security token) which conferred the subject confirmation key relied upon for sender authentication.
  The recipient MUST corroborate that the bound subject confirmation key is the same key used to authenticate the communicating peer.

- The recipient MUST determine that it trusts the authority which signed the `<saml:Assertion>`.
  The recipient MUST validate the signature of the `<saml:Assertion>`. The recipient SHOULD validate the trust semantics of the signing key, as appropriate to the risk of incorrect authentication.

8. Supporting Schema

This section describes the additional schema elements that support the authorization model described in Section 7.

8.1. ProxyTransitedStatement Schema

The <ProxyTransitedStatement> is used to identify an entity which actively participated in the message exchanges leading up to a given resource access. Its intended usage is twofold. First, the <ProxyTransitedStatement> MAY be used by a message recipient to convey to the assertion issuer subject confirmation data that was extracted from an assertion previously issued by that authority to the message sender. Second, the assertion issuing authority MAY use the <ProxyTransitedStatement> to propagate this information as advice within the assertion it subsequently generates and returns to the message recipient - this to be used within another resource access message.

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

```
<xs:element name="ProxyTransitedStatement" type="sec:ProxyTransitedStatementType"/>
 <xs:complexType name="ProxyTransitedStatementType">
  <xs:complexContent>
   <xs:extension base="saml:StatementAbstractType">
    <xs:sequence>
     <xs:element ref="saml:SubjectConfirmation" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
```

8.2. ProxyInfoConfirmationData Schema

A proxy uses the the <ProxyInfoConfirmationData> to supply subject confirmation data to an assertion issuer; this subject confirmation data previously used by another proxy in authenticating a message sent to the first proxy.

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

```
 <xs:complexType name="ProxyInfoConfirmationDataType" mixed="false">
  <xs:complexContent>
   <xs:restriction base="saml:SubjectConfirmationDataType">
    <xs:sequence>
     <xs:element ref="saml:AssertionIDRef"/>
     <xs:element ref="saml:Issuer" />
     <xs:element name="IssueInstant" type="xs:dateTime"/>
     <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID"/>
   </xs:restriction>
  </xs:complexContent>
 </xs:complexType>
```

The semantics around the elements are as follows:

- The <saml:AssertionIDRef>, <Issuer> and <IssueInstant> are that of the <saml:Assertion> presented by the proxy subject.
- The OPTIONAL <ds:Signature> element is a digital signature created by the recipient which covers the child elements of <ProxyInfoConfirmationData> with the exclusion of itself. It is RECOMMENDED that the enveloped signature transform (see [XMLDsig]) be utilized to accomplish the element exclusion.

9. Examples (Informative)

These examples demonstrate SAML 2.0 assertions. For examples that demonstrate SAML 1.1 assertions, as well as X.509 and Custom Bearer message authentication, refer to [LibertySecMech].

9.1. Fragmentary Examples

The examples in this section are fragments of full assertions - they are intended to demonstrate a particular aspect of the message syntax.

9.1.1. Sender as Invocation Identity

In the simplest of settings the sender of a message is acting on it's own behalf. The assertion issuing authority identifies the sender as the subject of the assertion.

```
001 <saml:Subject>
002    <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
003      http://ovaloffice.whitehouse.gov/</saml:NameID>
004   <saml:SubjectConfirmation
005       Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
006    <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
007
008      <!-- This keyinfo is the key by which the sender must
009         prove possession in order for the relying party to
010         accept the Statements in this assertion.  -->
011     <ds:KeyInfo>
012       <ds:KeyName>
013         CN=ovaloffice.whitehouse.gov,OU=Executive Branch,O=United States,...
014       </ds:KeyName>
015       <ds:KeyValue>...</ds:KeyValue>
016      </ds:KeyInfo>
017    </saml:SubjectConfirmationData>
018   </saml:SubjectConfirmation>
019 </saml:Subject>
```

Contents in the above example worth particular mention include lines 002-003 which specify the identifier is an entity id and the name of the sender. Lines 004-018 describe the confirmation requirements that the sender must uphold to be confirmed as the subject of the assertion. Line 005 mandates that the sender demonstrate possession of the confirmation key described in lines 011-016.

9.1.2. Sender as Proxy Identity

At times it is necessary to convey multiple identities to a relying party. One identity is the subject of the assertion. The other is that of a proxy which is acting on behalf of the subject. Typically the proxy is the sender of a message to a relying party and as such it's identity needs to be distinguished from that of the subject. To accomplish this the assertion issuer decorates the `saml:SubjectConfirmation` element with a `saml:NameID` element.

9.1.2.1. Transparent Subject Identifier

In the following example the identity of the subject is transparent to the proxy and the proxy is identified as the confirming entity.

```
001 <saml:Subject>
002   <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
003       president@whitehouse.gov</saml:NameID>
004   <saml:SubjectConfirmation
005       Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
006    <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
007      http://mailhost.whitehouse.gov/</saml:NameID>
```

```
008    <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
009
010      <!-- This keyinfo is the key by which the sender (aka proxy) must
011         prove possession in order for the relying party to
012         accept the Statements in this assertion.  -->
013      <ds:KeyInfo>
014        <ds:KeyName>
015          CN=mailhost.whitehouse.gov,OU=Executive Branch,O=United States,...
016        </ds:KeyName>
017        <ds:KeyValue>...</ds:KeyValue>
018       </ds:KeyInfo>
019    </saml:SubjectConfirmationData>
020   </saml:SubjectConfirmation>
021 </saml:Subject>
```

In the above example the noteworthy elements are described. Lines 002-003 describe the identity of the subject, aka the invocation identity. Lines 004-019 describe the confirmation requirements that the sender must uphold to be confirmed as the subject of the assertion. Line 005 mandates that the sender demonstrate possession of the confirmation key described in lines 008-020. Lines 006-007 identify the name of the proxy.

9.1.2.2. Opaque Subject Identifier Identifier

In the following example, the identity of the subject is made opaque to the proxy through encryption and the proxy is identified as the confirming entity.

```
001 <saml:Subject>
002   <saml:EncryptedID><xenc:EncryptedData>U2XTCNvRX7Bl1NK182nmY00TEk==</xenc:EncryptedData>
003       <xenc:EncryptedKey>...</xenc:EncryptedKey>
004   </saml:EncryptedID>
005   <saml:SubjectConfirmation
006       Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
007     <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
008       http://mailhost.whitehouse.gov/</saml:NameID>
009     <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
010
011       <!-- This keyinfo is the key by which the sender (aka proxy) must
012          prove possession in order for the relying party to
013          accept the Statements in this assertion.  -->
014       <ds:KeyInfo>
015         <ds:KeyName>
016           CN=mailhost.whitehouse.gov,OU=Executive Branch,O=United States,...
017         </ds:KeyName>
018         <ds:KeyValue>...</ds:KeyValue>
019        </ds:KeyInfo>
020     </saml:SubjectConfirmationData>
021   </saml:SubjectConfirmation>
022 </saml:Subject>
```

This example is very similar to the previous. The difference is that the name identifier for the subject of the assertion is encrypted, lines 002-004.

9.1.3. Invoking Identity Authentication

The relying party may need information regarding the authentication of the subject (aka invocation identity.) To accommodate this the assertion issuer decorates the assertion with an `<saml:AuthnStatement>`.

```
001 <!-- The saml:AuthnStatement carries information that
002     describes the authentication event of the subject
003     to an authenticating authority -->
004 <saml:AuthnStatement
```

```
005   AuthnInstant="2005-04-01T16:57:30.000Z"
006   SessionIndex="6345789">
007   <saml:AuthnContext>
008    <saml:AuthnContextClassRef>
009     urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
010    </saml:AuthnContextClassRef>
011   </saml:AuthnContext>
012 </saml:AuthnStatement>
```

Lines 005-006 describe attributes of the authentication event. Line 005 indicates the time at which authentication occurred. The session index between the subject and the authentication authority is on line 006. Lines 007-010 provide the technical details of the authentication action itself.

9.1.4. Resource as an Attribute

The assertion issuer may make coarse-grained authorization decisions and in so doing reflect precisely the resource for which the assertion is targeted. By identifying the resource in an attribute statement and binding the statement to the assertion the relying party can base it's authorization decision on the bound attribute and the actual resource being accessed. However, applications that use this specification may have alternative methods of referring to resources and thus disseminating this information in an attribute statement may be redundant.

9.1.4.1. Transparent Resource Identifier

In this example the Resource Identifier is transparent to the sender.

```
001  <saml:AttributeStatement>
002   <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
003              Name="urn:liberty:disco:2005-04:ResourceID">
004    <saml:AttributeValue xsi:type="disco:ResourceID>
005     http://wsp.example.com/pp?id=foobar</saml:AttributeValue>
006   </saml:Attribute>
007  </saml:AttributeStatement>
```

9.1.4.2. Opaque Resource Identifier

In operational settings which require opacity of identifiers (i.e. due to privacy requirements) then the attributes would be encrypted and packaged in a `<saml:EncryptedAttribute>` as is shown from lines 006-019 in the example below.

```
001 <!-- The AttributeStatement carries an EncryptedAttribute.
002     Once this element is decrypted with the supplied key
003     an <Attribute> element bearing an <disco:ResourceID>
004     can be found.  -->
005 <saml:AttributeStatement>
006   <saml:EncryptedAttribute>
007    <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
008    mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
009    YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpM+sF2BnpND118f/mXCv3XbWhiL
010    xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
011    ...
012    VRu9BpYBD4Y/98y1jtX9Pm898+xzketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
013    hg6nZ5c0I6L6Gn9A
014    =HCQY
015    </xenc:EncryptedData>
016    <xenc:EncryptedKey>
017      ...
018    </xenc:EncryptedKey>
019   </saml:EncryptedAttribute>
```

```
020 </saml:AttributeStatement>
```

9.2. Proxying with Authentication Context of the Invoking Identity

Access to resources exposed by a service instance are nominally restricted by access control policy enforced by the entity hosting the resource. Additionally, the policy information, enforcement and decision points may be distributed across multiple system entities. Authorization to access a resource may require that the entity interacting (e.g. browser principal) with another entity (e.g. service consumer) have an active authenticated session.

To facilitate this scenario the trusted authority may supply authorization data that conveys the session status of the interacting entity. This is accomplished by including a `<saml:AuthnStatement>` in the assertion.

The following example demonstrates:

- Proxying
- Encrypted Name Identifier
- Encrypted Resource Identifier

```
<?xml version="1.0" encoding="UTF-8"?>
 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:sb="urn:liberty:sb:2003-08"
          xmlns:pp="urn:liberty:id-sis-pp:2003-08"
          xmlns:sec="urn:liberty:sec:2004-10"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
ity-secext-1.0.xsd">


  <s:Header>
     <sb:Correlation s:mustUnderstand="1"
                id="A13454...245"
                actor="http://schemas.../next"
                messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
                timestamp="2112-03-15T11:12:12Z"/>
   <wsse:Security>
     <saml:Assertion
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      Version="2.0"
      ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
      IssueInstant="2005-04-01T16:58:33.173Z">

      <saml:Issuer>http://authority.example.com/</saml:Issuer>

      <!-- signature by the issuer over the assertion -->
      <ds:Signature>...</ds:Signature>

      <!-- By placing an audience restriction on the assertion we
      can limit the scope of which entity should consume
      the information in the assertion. -->

      <saml:Conditions
        NotBefore="2005-04-01T16:57:20Z"
        NotOnOrAfter="2005-04-01T21:42:43Z">

        <saml:AudienceRestrictionCondition>
          <saml:Audience>http://wsp.example.com</saml:Audience>
        </saml:AudienceRestrictionCondition>
      </saml:Conditions>

      <saml:Subject>
```

```
      <saml:EncryptedID>
       <xenc:EncryptedData>U2XTCNvRX7Bl1NK182nmY00TEk==</xenc:EncryptedData>
       <xenc:EncryptedKey>...</xenc:EncryptedKey>
      </saml:EncryptedID>

      <saml:SubjectConfirmation
       Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
       <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
          http://wsc.example.com/</saml:NameID>
       <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">

         <!-- This keyinfo is the key by which the sender must
             prove possession in order for the relying party to
             accept the Statements in this assertion.  -->
         <ds:KeyInfo>
           <ds:KeyName>
             CN=wsc.example.com,OU=Client Services R US,O=Service Station,...
           </ds:KeyName>
            <ds:KeyValue>...</ds:KeyValue>
         </ds:KeyInfo>
       </saml:SubjectConfirmationData>
      </saml:SubjectConfirmation>
     </saml:Subject>

     <!-- The AuthnStatement carries information
         that describes the authentication event
         of the Subject to an Authentication Authority -->
     <saml:AuthnStatement
      AuthnInstant="2005-04-01T16:57:30.000Z"
      SessionIndex="6345789">
      <saml:AuthnContext>
       <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
       </saml:AuthnContextClassRef>
      </saml:AuthnContext>
     </saml:AuthnStatement>

     <!-- The AttributeStatement carries an EncrpytedAttribute.
         Once this element is decrypted with the supplied key
         an <Attribute> element bearing an <disco:ResourceID>
         can be found.  -->
     <saml:AttributeStatement>
      <saml:EncryptedAttribute>
       <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
         mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
         YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpM+sF2BnpND118f/mXCv3XbWhiL
         xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
         ...
         VRu9BpYBD4Y/98y1jtX9Pm898+xzketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
         hg6nZ5c0I6L6Gn9A
         =HCQY
       </xenc:EncryptedData>
       <xenc:EncryptedKey> ... </xenc:EncryptedKey>
      </saml:EncryptedAttribute>
     </saml:AttributeStatement>

    </saml:Assertion>
    <!-- this is the signature the sender generated to demonstrate holder-of-key
        the signature should cover the isf header and body-->
    <ds:Signature>
     <ds:SignedInfo>
       ...
       <ds:Reference URI="#A13454...245">
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
       </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:KeyInfo>
       <wsse:SecurityTokenReference>
        <wsse:KeyIdentifier
         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-1.
0#SAMLAssertionID" />
         2sxJu9g/vvLG9sAN9bKp/8q0NKU=
        </wsse:KeyIdentifier>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
    </ds:SignatureValue>
   </ds:Signature>
  </wsse:Security>
   </s:Header>
   <s:Body id="MsgBody">
    <pp:Modify>
         <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
   </s:Body>
</s:Envelope>
```

9.3. Conveyance of Sender as Invocation Identity

This example depicts a request to access an identity-based web service in which the sender identity and the invocation identity are the same (i.e. non-proxsying). The resource which the sender is attempting to access is described in an `<AttributeStatement>` within the assertion.

Note that, while the assertion associates a subject's name with a key, this association is made as a means to indicate the authorization of that subject, acting with that key, to invoke a service. This facility, incorporated for authorization purposes, serves a distinct and complementary function to the binding between subject and key, which the subject's certificate accomplishes for authentication purposes.

The example demonstrates:

- Sender is Invocation Identity.
- Transparent Resource Identifier.

```
<?xml version="1.0" encoding="UTF-8"?>
 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:sb="urn:liberty:sb:2003-08"
         xmlns:pp="urn:liberty:id-sis-pp:2003-08"
         xmlns:sec="urn:liberty:sec:2004-10"
         xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
ity-secext-1.0.xsd">


  <s:Header>
     <sb:Correlation s:mustUnderstand="1"
                id="A13454...245"
                actor="http://schemas.../next"
                messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
                timestamp="2112-03-15T11:12:12Z"/>
   <wsse:Security>
    <saml:Assertion
     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
     Version="2.0"
     ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
```

```
      IssueInstant="2005-04-01T16:58:33.173Z">

      <saml:Issuer>http://authority.example.com/</saml:Issuer>

      <!-- signature by the issuer over the assertion -->
      <ds:Signature>...</ds:Signature>

      <!-- By placing an audience restriction on the assertion we
      can limit the scope of which entity should consume
      the information in the assertion. -->

      <saml:Conditions
        NotBefore="2005-04-01T16:57:20Z"
        NotOnOrAfter="2005-04-01T21:42:43Z">

        <saml:AudienceRestrictionCondition>
          <saml:Audience>http://wsp.example.com</saml:Audience>
        </saml:AudienceRestrictionCondition>
      </saml:Conditions>


      <saml:Subject>
         <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
          http://ovaloffice.whitehouse.gov/</saml:NameID>
        <saml:SubjectConfirmation
            Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
          <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">

            <!-- This keyinfo is the key by which the sender must
                prove possession in order for the relying party to
                accept the Statements in this assertion.  -->
            <ds:KeyInfo>
              <ds:KeyName>
               CN=ovaloffice.whitehouse.gov,OU=Executive Branch,O=United States,...
              </ds:KeyName>
              <ds:KeyValue>...</ds:KeyValue>
             </ds:KeyInfo>
          </saml:SubjectConfirmationData>
        </saml:SubjectConfirmation>
      </saml:Subject>


      <saml:AttributeStatement>
        <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                   Name="urn:liberty:disco:2005-04:ResourceID">
          <saml:AttributeValue xsi:type="disco:ResourceID>
            http://wsp.example.com/pp?id=foobar</saml:AttributeValue>
        </saml:Attribute>
      </saml:AttributeStatement>
    </saml:Assertion>
    <!-- this is the signature the sender generated to demonstrate holder-of-key
        the signature should cover the isf header and body-->
    <ds:Signature>
     <ds:SignedInfo>
       ...
       <ds:Reference URI="#A13454...245">
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
       </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:KeyInfo>
       <wsse:SecurityTokenReference>
         <wsse:KeyIdentifier
```

```
         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004
XX-wss-saml-token-profile-1.0#SAMLAssertionID" />
         2sxJu9g/vvLG9sAN9bKp/8q0NKU=
        </wsse:KeyIdentifier>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
    </ds:SignatureValue>
   </ds:Signature>
  </wsse:Security>
   </s:Header>
   <s:Body id="MsgBody">
    <pp:Modify>
         <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
   </s:Body>
</s:Envelope>
```

9.4. Proxy Chaining

The following example demonstrates:

- Proxy Chain captured in `<ProxyTransitedStatement>` as multiple `<SubjectConfirmation>` elements. Two different proxies separate from the sender are listed.
- Encrypted Name Identifier.
- Encrypted Resource Identifier.
- Authentication status of Invoking Identity.

```
<?xml version="1.0" encoding="UTF-8"?>
 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:sb="urn:liberty:sb:2004-04"
         xmlns:pp="urn:liberty:id-sis-pp:2003-08"
         xmlns:sec="urn:liberty:sec:2004-10"
         xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
rity-secext-1.0.xsd">


  <s:Header>
     <sb:Correlation s:mustUnderstand="1"
                id="A13454...245"
                actor="http://schemas.../next"
                messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
                timestamp="2112-03-15T11:12:12Z"/>
   <wsse:Security>
    <saml:Assertion
     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
     Version="2.0"
     ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
     IssueInstant="2005-04-01T16:58:33.173Z">

     <saml:Issuer>http://authority.example.com/</saml:Issuer>

     <!-- signature by the issuer over the assertion -->
     <ds:Signature>...</ds:Signature>

     <saml:Advice>
      <saml:AssertionIDRef>refers to this assertion</saml:AssertionIDReference>
      <saml:Assertion>
```

```
        <!-- This statement reflects path of proxy transitions The
        list is comprised of SubjectConfirmation elements in the
        order the proxy was transitioned (first to last). -->

        <sec:ProxyTransitedStatement>
         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
          <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
           http://first.example.com/</saml:NameID>
          <saml:SubjectConfirmationData xsi:type="sec:ProxyInfoConfirmationDataType">
            <saml:AssertionIDRef>
<!-- refers to an assertion issued by the assertion issuer to first.example.com. -->
            </saml:AssertionIDRef>
            <saml:Issuer>authority.example.com</saml:Issuer>
            <sec:IssueInstant>2004-04-01T16:58:30.173Z</sec:IssueInstant>
          </saml:SubjectConfirmationData>
       </sec:SubjectConfirmation>

         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
          <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
           http://second.example.com/</saml:NameID>
          <saml:SubjectConfirmationData xsi:type="sec:ProxyInfoConfirmationDataType">
            <saml:AssertionIDRef>
<!-- refers to an assertion issued by the assertion issuer to second.example.com. -->
            </saml:AssertionIDRef>
            <saml:Issuer>authority.example.com</saml:Issuer>
            <sec:IssueInstant>2004-04-01T16:58:40.173Z</sec:IssueInstant>
          </saml:SubjectConfirmationData>
       </sec:SubjectConfirmation>
        </sec:ProxyTransitedStatement>
      </saml:Assertion>
    </saml:Advice>

    <!-- By placing an audience restriction on the assertion we
    can limit the scope of which entity should consume
    the information in the assertion. -->

    <saml:Conditions
     NotBefore="2005-04-01T16:57:20Z"
     NotOnOrAfter="2005-04-01T21:42:43Z">

      <saml:AudienceRestrictionCondition>
       <saml:Audience>http://wsp.example.com</saml:Audience>
      </saml:AudienceRestrictionCondition>
    </saml:Conditions>

    <saml:Subject>
     <saml:EncryptedID>
      <xenc:EncryptedData>U2XTCNvRX7Bl1NK182nmY00TEk==</xenc:EncryptedData>
      <xenc:EncryptedKey>...</xenc:EncryptedKey>
     </saml:EncryptedID>

      <saml:SubjectConfirmation
       Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
       <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
          http://third.example.com/</saml:NameID>
       <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">

         <!-- This keyinfo is the key by which the sender must
             prove possession in order for the relying party to
             accept the Statements in this assertion.  -->
         <ds:KeyInfo>
           <ds:KeyName>
             CN=third.example.com,OU=Client Services R US,O=Service Station,...
           </ds:KeyName>
            <ds:KeyValue>...</ds:KeyValue>
         </ds:KeyInfo>
```

```
        </saml:SubjectConfirmationData>
      </saml:SubjectConfirmation>
    </saml:Subject>

    <!-- The AuthnStatement carries information
        that describes the authentication event
        of the Subject to an Authentication Authority -->
    <saml:AuthnStatement
      AuthnInstant="2005-04-01T16:57:30.000Z"
      SessionIndex="6345789">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
          urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>

    <!-- The AttributeStatement carries an EncryptedAttribute.
        Once this element is decrypted with the supplied key
        an <Attribute> element bearing an <disco:ResourceID>
        can be found.  -->
    <saml:AttributeStatement>
      <saml:EncryptedAttribute>
        <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
          mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
          YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpM+sF2BnpND118f/mXCv3XbWhiL
          xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
          ...
          VRu9BpYBD4Y/98y1jtX9Pm898+xzketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
          hg6nZ5c0I6L6Gn9A
          =HCQY
        </xenc:EncryptedData>
        <xenc:EncryptedKey> ... </xenc:EncryptedKey>
      </saml:EncryptedAttribute>
    </saml:AttributeStatement>

  </saml:Assertion>
  <!-- this is the signature the sender generated to demonstrate holder-of-key
      the signature should cover the header and body-->
  <ds:Signature>
   <ds:SignedInfo>
     ...
     <ds:Reference URI="#A13454...245">
       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
     </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:KeyInfo>
     <wsse:SecurityTokenReference>
       <wsse:KeyIdentifier
        ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-t
oken-profile-1.0#SAMLAssertionID" />
        2sxJu9g/vvLG9sAN9bKp/8q0NKU=
       </wsse:KeyIdentifier>
   </ds:KeyInfo>
   <ds:SignatureValue>
    HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
  </ds:SignatureValue>
 </ds:Signature>
</wsse:Security>
 </s:Header>
 <s:Body id="MsgBody">
  <pp:Modify>
```

```
          <!-- this is an ID-SIS-PP Modify message -->
     </pp:Modify>
   </s:Body>
</s:Envelope>
```

9.5. SAML Bearer Token

The following example demonstrates the Bearer message authentication mechanism by supplying a SAML bearer token [wss-saml] in the security header.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:sb="urn:liberty:sb:2003-08"
         xmlns:pp="urn:liberty:id-sis-pp:2003-08"
         xmlns:sec="urn:liberty:sec:2004-10"
         xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">

  <s:Header>
    <sb:Correlation s:mustUnderstand="1"
                id="A13454...245"
                actor="http://schemas.../next"
                messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
                timestamp="2112-03-15T11:12:12Z"/>

    <wsse:Security>
     <!-- this is an embedded reference to the bearer token -->
     <wsse:SecurityTokenReference>
      <wsse:Embedded
       ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-p
rofile-1.0#SAMLAssertionID">

       <saml:Assertion
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        Version="2.0"
        ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
        IssueInstant="2005-04-01T16:58:33.173Z">

        <saml:Issuer>http://authority.example.com/</saml:Issuer>

        <!-- signature by the issuer over the assertion -->
        <ds:Signature>...</ds:Signature>

        <!-- By placing an audience restriction on the assertion we
        can limit the scope of which entity should consume
        the information in the assertion. -->

        <saml:Conditions
         NotBefore="2005-04-01T16:57:20Z"
         NotOnOrAfter="2005-04-01T21:42:43Z">

         <saml:AudienceRestrictionCondition>
          <saml:Audience>http://wsp.example.com</saml:Audience>
         </saml:AudienceRestrictionCondition>
        </saml:Conditions>

        <saml:Subject>
         <saml:EncryptedID>
          <xenc:EncryptedData>U2XTCNvRX7Bl1NK182nmY00TEk==</xenc:EncryptedData>
          <xenc:EncryptedKey>...</xenc:EncryptedKey>
         </saml:EncryptedID>

         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
         </saml:SubjectConfirmation>
```

```
        </saml:Subject>

        <!-- The AuthnStatement carries information
            that describes the authentication event
            of the Subject to an Authentication Authority -->
        <saml:AuthnStatement
          AuthnInstant="2005-04-01T16:57:30.000Z"
          SessionIndex="6345789">
          <saml:AuthnContext>
            <saml:AuthnContextClassRef>
              urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
            </saml:AuthnContextClassRef>
          </saml:AuthnContext>
        </saml:AuthnStatement>

        <!-- The AttributeStatement carries an EncrpytedAttribute.
            Once this element is decrypted with the supplied key
            an <Attribute> element bearing an <disco:ResourceID>
            can be found.  -->
        <saml:AttributeStatement>
          <saml:EncryptedAttribute>
            <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
              mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
              YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpM+sF2BnpND118f/mXCv3XbWhiL
              xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
              ...
              VRu9BpYBD4Y/98y1jtX9Pm898+xzketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
              hg6nZ5c0I6L6Gn9A
              =HCQY
            </xenc:EncryptedData>
            <xenc:EncryptedKey> ... </xenc:EncryptedKey>
          </saml:EncryptedAttribute>
        </saml:AttributeStatement>

      </saml:Assertion>
    </wsse:Embedded>
   </wsse:SecurityTokenReference>
  </wsse:Security>
 </s:Header>
 <s:Body id="MsgBody">
    <pp:Modify>
         <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
 </s:Body>
</s:Envelope>
```

9.6. X.509 v3 Message Authentication

The following example demonstrates 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:2003-08"
          xmlns:pp="urn:liberty:id-sis-pp:2003-08"
          xmlns:sec="urn:liberty:sec:2003-08"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
ssecurity-secext-1.0.xsd">


   <s:Header>
    <sb:Correlation s:mustUnderstand="1"
                id="A13454...245"
                actor="http://schemas.../next"
                messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
```

```
                timestamp="2112-03-15T11:12:12Z"/>
    <wsse:Security xmlns:wsse="...">
      <wsse:BinarySecurityToken ValueType="wsse:X509v3" wsu:Id="X509Token"
                        EncodingType="wsse:Base64Binary">
        MIIB9zCCAWSgAwIBAgIQ...
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>

          <!-- bind the correlation header ->
          <ds:Reference URI="#A13454...245">
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
          </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 id="MsgBody">
    <pp:Modify>
         <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
  </s:Body>
</s:Envelope>
```

9.7. Custom Bearer Token Message Authentication

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 be able to determine the proper processing rules based off of 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:2003-08"
        xmlns:pp="urn:liberty:id-sis-pp:2003-08"
        xmlns:sec="urn:liberty:sec:2004-10"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">

 <s:Header>
   <sb:Correlation s:mustUnderstand="1"
               id="A13454...245"
               actor="http://schemas.../next"
               messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
               timestamp="2112-03-15T11:12:12Z"/>
```

```
    <wsse:Security>
     <!-- Custom binary security token -->
     <wsse:BinarySecurityToken
         ValueType="anyNSPrefix:ServiceSessionContext"
         EncodingType="wsse: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>

     <!-- this is the reference to the above bearer token -->
     <wsse:SecurityTokenReference>
        <wsse:Reference URI="#bst" />
     </wsse:SecurityTokenReference>
    </wsse:Security>
  </s:Header>
  <s:Body id="MsgBody">
     <!-- payload -->
  </s:Body>
</s:Envelope>
```

10. Schema

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

<xs:schema targetNamespace="urn:liberty:sec:2004-12"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        xmlns:disco="urn:liberty:disco:2004-12"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:sec="urn:liberty:sec:2004-12"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:md="urn:liberty:metadata:2004-12"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
   <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
           schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
   <xs:import namespace="urn:liberty:disco:2004-12"
           schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
<!--   <xs:import namespace="urn:liberty:ac:2004-12"
           schemaLocation="liberty-authentication-context-v2.0.xsd"/> -->
   <xs:import namespace="urn:liberty:metadata:2004-12"
           schemaLocation="liberty-metadata-v2.0.xsd"/>
   <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"
           schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
           schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-co
re-schema.xsd"/>
   <xs:annotation>
      <xs:documentation>Liberty ID-WSF Security Mechanisms Specification XSD</xs:documentation>
      <xs:documentation>
The source code in this XSD file was excerpted verbatim from:

Liberty ID-WSF Security Mechanisms Specification
Version 2.0-03
22 November 2004

        Copyright (c) 2004 Liberty Alliance participants, see
        http://www.projectliberty.org/specs/idwsf_2_0_copyrights.php

      </xs:documentation>
   </xs:annotation>

  <xs:element name="ProxyTransitedStatement" type="sec:ProxyTransitedStatementType"/>
   <xs:complexType name="ProxyTransitedStatementType">
     <xs:complexContent>
       <xs:extension base="saml:StatementAbstractType">
         <xs:sequence>
           <xs:element ref="saml:SubjectConfirmation" minOccurs="1" maxOccurs="unbounded"/>
         </xs:sequence>
       </xs:extension>
     </xs:complexContent>
   </xs:complexType>   <xs:complexType name="ProxyInfoConfirmationDataType" mixed="false">
     <xs:complexContent>
       <xs:restriction base="saml:SubjectConfirmationDataType">
         <xs:sequence>
           <xs:element ref="saml:AssertionIDRef"/>
           <xs:element ref="saml:Issuer" />
           <xs:element name="IssueInstant" type="xs:dateTime"/>
           <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
         <xs:attribute name="id" type="xs:ID"/>
       </xs:restriction>
     </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

Bibliography

Normative References

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

[LibertySecMechV20] Ellison, Gary, Madsen, Paul, eds. "Liberty ID-WSF Security Mechanisms," Version 2.0-03, Liberty Alliance Project (22 November 2004). *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*

[LibertyBindProf] Cantor, Scott, Kemp, John, Champagne, Darryl, eds. "Liberty ID-FF Bindings and Profiles Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[LibertyDisco] Beatty, John, Hodges, Jeff, Sergent, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0-02, Liberty Alliance Project (24 Nov 2004). *http://www.projectliberty.org/specs*

[LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 2.0-02, Liberty Alliance Project (25 November 2004). *http://www.projectliberty.org/specs*

[LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. " Liberty ID-WSF SOAP Binding Specification ," Version 2.0-01, Liberty Alliance Project (22 November 2004). *http://www.projectliberty.org/specs*

[LibertyGlossary] Hodges, Jeff, eds. "Liberty Technical Glossary," Version 1.3-errata-v1.0, Liberty Alliance Project (12 Aug 2004). *http://www.projectliberty.org/specs*

[SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (24 September 2004). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," OASIS Committee Draft 02, v2.0 , Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/security/*

[SAMLBind11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[SAMLBind2] Hughes, John, Cantor, Scott, Mishra, Prateek, Hirsch, Frederick, Philpott, Rob, Hodges, Jeff, Maler, Eve, eds. (18 August 2004). "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0," OASIS Committee Specification, version 2.0, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf*

[wss-saml] 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/1911/WSS-SAML-07.pdf* "Web Services Security: SAML Token Profile," Draft WSS-SAML-07.pdf,

[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 "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. (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 [June 2002]*.

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

[RFC2246] Dierks, T., Allen, C., eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2246.txt [January 1999]*.

[Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlschema-1/*