Liberty ID-WSF Security Mechanisms

Version: 1.2

Editors:
Gary Ellison, Sun Microsystems, Inc.

Contributors:
Robert Aarts, Nokia Corporation
Carolina Canales-Valenzuela, Ericsson
Scott Cantor, Internet2, The Ohio State University
Frederick Hirsch, Nokia Corporation
Jeff Hodges, Sun Microsystems, Inc.
John Kemp, IEEE-ISTO
John Linn, RSA Security Inc.
Paul Madsen, Entrust, 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: liberty-idwsf-security-mechanisms-v1.2.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

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 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 WS-Security ([wss-sms]) compliant header elements communicate the relevant security information, i.e., a SAML [SAMLCore11] 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 either retrieve information about an identity, update information related to an identity, or to 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 a 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:2003-08*
sb	*urn:liberty:sb:2003-08*
disco	*urn:liberty:disco:2003-08*
ac	*urn:liberty:ac:2003-08*
lib	*urn:liberty:iff:2003-08*
md	*urn:liberty:metadata:2003-08*
saml	*urn:oasis:names:tc:SAML:1.0: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

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. Message 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 URI (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 URI, 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:NameIdentifier>` following the processing rules in the NameIdentifier Encryption Profile specified in [LibertyBindProf].

5.3.2. Encrypted URI

At times it may be necessary to privacy protect the contents of a URI to deter the release of sensitive information to an intermediary. The [LibertyDisco] specification defines an encrypted form of a URI with the `<disco:EncryptedResourceID>` schema element. This specification relies upon the semantics defined in [LibertyDisco] to fulfill this privacy requirement. Thus the processing rules defined by [LibertyDisco] for the `<disco:EncryptedResourceID>` 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 the authentication mechanism identifiers and their security properties. Each URI is of the form *urn:liberty:security:date:peer mechanism:message mechanism*.

Table 2. Authentication Mechanisms

URI	Peer Entity	Message
urn:liberty:security:2003-08:null:null	No	No
urn:liberty:security:2005-02:null:X509	No	Yes
urn:liberty:security:2005-02:null:SAML	No	Yes
urn:liberty:security:2005-02:null:Bearer	No	No
urn:liberty:security:2003-08:TLS:null	Recipient	No
urn:liberty:security:2005-02:TLS:X509	Recipient	Yes
urn:liberty:security:2005-02:TLS:SAML	Recipient	Yes
urn:liberty:security:2005-02:TLS:Bearer	Recipient	No
urn:liberty:security:2003-08:ClientTLS:null	Mutual	No
urn:liberty:security:2005-02:ClientTLS:X509	Mutual	Yes
urn:liberty:security:2005-02:ClientTLS:SAML	Mutual	Yes
urn:liberty:security:2005-02:ClientTLS:Bearer	Mutual	No

6.1. Authentication Mechanism Overview (Informative)

The above table depicts the various authentication mechanism identifiers and the security 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 basically 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 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 making 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. The *Bearer* mechanism (Section 6.3.3) is based on the presence of a *bearer token* in the security header of a message. 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:2005-02:ClientTLS:X509* is relevant. However, such a combination may make little sense when peer entity authentication is relied upon to imply message authentication. For example, the mechanism *urn:liberty:security:2005-02:ClientTLS:X509* seems equivalent to *urn:liberty:security:2003-08:ClientTLS:null* in such a setting. A similar argument can be made for the *urn:liberty:security:2005-02:ClientTLS:SAML* 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 rely upon the inherent security properties of the TLS/SSL protocol (sometimes referred to as transport-level security). 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 the signature attesting to the authenticity of the message need not 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.

6.2.1. Unilateral Peer Entity Authentication

The following URIs support unilateral (recipient) peer entity authentication:

- *urn:liberty:security:2003-08:TLS:null*
- *urn:liberty:security:2005-02:TLS:X509*
- *urn:liberty:security:2005-02:TLS:SAML*
- *urn:liberty:security:2005-02: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 following URIs support mutual (sender and recipient) peer entity authentication:

- *urn:liberty:security:2003-08:ClientTLS:null*
- *urn:liberty:security:2005-02:ClientTLS:X509*
- *urn:liberty:security:2005-02:ClientTLS:SAML*
- *urn:liberty:security:2005-02: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 message authentication mechanisms prescribed by this specification rely upon the integrity properties imbued by the application and verification of digital signatures over elements of the message 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 the signature attesting to the authenticity of the message need not be retained, then it may be sufficient to rely upon the properties of peer entity authentication to assure the integrity and authenticity of the message payload. Another example might be when unilateral peer entity authentication of the recipient can be combined with message authentication of the sender such that the response could be deemed trustworthy in the absence of a signature over the payload of said response.

6.3.1. X.509 v3 Certificate Message Authentication

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

- *urn:liberty:security:2005-02:null:X509*
- *urn:liberty:security:2005-02:TLS:X509*
- *urn:liberty:security:2005-02: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 author 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. 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 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 security token. 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:2005-02: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 carrying 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>`.

6.3.1.3. Example X.509 v3 Message Authentication (Informative)

The following example demonstrates this 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-ws
security-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="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x
509-token-profile-1.0#X509v3"
          wsu:Id="X509Token"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#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 wsu:Id="MsgBody">
    <pp:Modify>
         <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
   </s:Body>
</s:Envelope>
```

6.3.2. SAML Assertion Message Authentication

The following URIs indicate SAML-based unilateral message-origin (sender) message authentication mechanisms:

- *urn:liberty:security:2005-02:null:SAML*
- *urn:liberty:security:2005-02:TLS:SAML*
- *urn:liberty:security:2005-02:ClientTLS:SAML*

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:AuthenticationStatement>` and other statements derived from `<saml:SubjectStatement>`. The `<saml:AuthenticationStatement>` describes the authentication event at the issuing authority (a TTP.) For each of the `<saml:SubjectStatement>` bound into the assertion, the authority dictates the subject confirmation obligations the subject must affirm to a relying party in order for the statement to be considered trustworthy.

As a security precaution, the issuer of the assertion MUST include a `<saml:AudienceRestrictionCondition>` element that specifies the intended consumer(s) of the assertion. One `<saml:Audience>` element MUST be set to the intended recipient's `<ProviderID>` as specified in [LibertyMetadata]. The recipient MUST validate that it is the intended consumer before using the assertion. The assertion MAY contain additional `<saml:Audience>` elements that specify other intended relying parties.

These message authentication mechanisms are unilateral. That is only, the author 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 would offer.

For deployment settings which 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. This key MUST be recognized by the recipient as belonging to the sender.

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 `<wsse:Security>`.

- The sender MUST adhere to its subject confirmation obligation in accordance with the semantics of the confirmation method described by each <saml:SubjectStatement> bound into the <saml:Assertion>.
  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 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 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. This 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 signature of 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 the confirmation obligation of each <saml:SubjectStatement> bound within the assertion. Only the statements which comply with the confirmation obligation bound into the statement should be considered trustworthy.

- 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:2005-02:null:Bearer*
- *urn:liberty:security:2005-02:TLS:Bearer*
- *urn:liberty:security:2005-02: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/oasis-wss-saml-token-profile-1.0#SAMLAssertionID` [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` [SAMLBind11] and that the relying party would validate the assertion in accordance with the processing rules of [SAMLCore11]. Particular attention must be paid to the proper validation of the `<saml:AudienceRestrictionCondition>` 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 9.4.

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 second example in Section 9.4.

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].

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

- When the token type is not well known or in an obvious form, 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.

- This bearer token MUST NOT be referenced by a WS-Security signature in the message.

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. Supporting Schema

The authorization mechanism defined in this specification relies upon supporting XML Schema. The defined schema fosters the conveyance of authorization information within a given message exchange. This section defines new schema types as well as the utilization of schema elements defined in other specifications.

7.1. ProxySubject Schema

The <ProxySubject> is used to convey the identity of a proxy, the confirmation key and confirmation obligation the proxy must possess and demonstrate for authentication purposes. The following schema fragment describes the structure of the <ProxySubject> element:

```
   <xs:element name="ProxySubject" substitutionGroup="saml:Subject"
            type="saml:SubjectType"/>
```

The construction of a <ProxySubject> element MUST adhere to the following constraints:

- The <ProxySubject> SHOULD include a <saml:SubjectConfirmation> element with a <saml:ConfirmationMethod> of urn:oasis:names:tc:SAML:1.0:cm:holder-of-key.
- The <ProxySubject> SHOULD specify the subject confirmation key by including a <ds:KeyInfo> element as a child of the <saml:SubjectConfirmation> element.

7.2. 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, it MAY be used by the consumer of authorization data for conveyance to the assertion issuing authority. Second, the assertion issuing authority MAY propagate this information as advice within assertions it subsequently generates.

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

```
   <xs:annotation>
      <xs:documentation>ProxyTransitedStatement is a
      SubjectStatement which MAY carry specific subject confirmation
      data </xs:documentation>
   </xs:annotation>
   <xs:element name="ProxyTransitedStatement"
 type="saml:SubjectStatementAbstractType"/>
```

7.3. ProxyInfoConfirmationData Schema

The <ProxyInfoConfirmationData> is used to supply subject confirmation data which was demonstrated by a system entity to a trusted authority which resulted in propagating a resource access to a proxy.

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

```
   <xs:annotation>
      <xs:documentation>
        ProxyInfoConfirmationData may be relied upon to
        corroborate the path information carried in a
        ProxyTransitedStatement
 </xs:documentation>
```

```
  </xs:annotation>
  <xs:element name="ProxyInfoConfirmationData"
          type="sec:ProxyInfoConfirmationType"/>
  <xs:complexType name="ProxyInfoConfirmationType">
    <xs:sequence>
      <xs:element ref="saml:AssertionIDReference"/>
      <xs:element name="Issuer" type="xs:string"/>
      <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:complexType>
```

The semantics around the elements are as follows:

- The <saml:AssertionIDReference>, <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.

7.4. SessionContext Schema

The <SessionContext> element conveys session status of an entity to another system entity. In general it is supplied to a relying party to support policy enforcement.

The following schema fragment describes the structure of the <SessionContext> element:

```
  <xs:element name="SessionContext" type="sec:SessionContextType"/>
  <xs:complexType name="SessionContextType">
     <xs:sequence>
        <xs:element name="SessionSubject" type="lib:SubjectType"/>
        <xs:element name="ProviderID" type="md:entityIDType"/>
        <xs:element ref="lib:RequestAuthnContext" minOccurs="0" maxOccurs="1"/>
        <!-- The system entity for which this context applies
    is privacy protect by the SessionSubject -->
     </xs:sequence>
     <xs:attribute name="SessionIndex" type="xs:string" use="optional"/>
     <xs:attribute name="AuthenticationInstant" type="xs:dateTime" use="required"/>
     <xs:attribute name="AssertionIssueInstant" type="xs:dateTime" use="required"/>
  </xs:complexType>
```

The contents of the <SessionContext> element MUST adhere to the following constraints:

- The <SessionSubject> element could be available to parties which are not privy to the name identifier of the subject. When this is the case the <SessionSubject> element MUST be constructed in such a way as to protect the privacy of the identifier it carries by adhering to the rules defined in Encrypted Name Identifiers (Section 5.3.1). An example of when the contents would not require these privacy protections is when the name identifier in the SessionSubject is produced and consumed by the same entity and the party with which it passes through has knowledge of the name.
  The identifier MUST indicate (when decrypted) the identity of the subject with which the session applies. This identity MUST be resolvable in the namespace of the entity consuming the <SessionContext> element.

- The `<ProviderID>` element is a URI which indicates the entity (session holder) with which the session applies. `<ProviderID>` MUST correspond with the constraints for the `<entityIDType>` data type as specified in [LibertyMetadata].

- The `<ac:AuthnContext>` element as specified in [LibertyAuthnContext] depicts the authentication context which the session holder relied upon to create the session.

- The `SessionIndex` attribute is used as an aid when in managing multiple sessions with a Principal. This attribute represents the same thing as the attribute of the same name in the `lib:AuthenticationStatement` [LibertyProtSchema]

- The `AuthenticationInstant` attribute describes the time at which the subject of the session authenticated to an authority.

- The `AssertionIssueInstant` attribute is the time at which the assertion issuer generated an assertion for the provider.

7.5. SessionContextStatement Schema

The `<SessionContextStatement>` element conveys session status of an entity to another system entity within the body of an `<saml:Assertion>`.

The following schema fragment describes the structure of the `<SessionContextStatement>` element:

```
<xs:element name="SessionContextStatement"
        type="sec:SessionContextStatementType"
        substitutionGroup="saml:SubjectStatement"/>

<xs:complexType name="SessionContextStatementType">
   <xs:complexContent>
      <xs:extension base="saml:SubjectStatementAbstractType">
         <xs:sequence>
            <!-- This is the name of the proxy and it SHOULD carry
                SubjectConfirmation information to authorize the
                ProxySubject to act on behalf of the
                Subject inherited from
                SubjectStatementAbstractType -->
            <xs:element name="ProxySubject"
                    type="saml:SubjectType"  minOccurs="0"/>
            <xs:element ref="sec:SessionContext"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The `<SessionContextStatement>` derives from `<saml:SubjectStatementAbstractType>` and MUST adhere to the following constraints:

- It MAY include a `<ProxySubject>` element which adheres to the structure and semantics described in this specification

- It MUST include a `<SessionContext>` element which adheres to the structure and semantics described in this specification

7.6. ResourceAccessStatement Schema

Resource access information is captured in a <ResourceAccessStatement> element. The purpose of this statement is to convey sufficient information regarding the accessing entity and the resource for which access is being attempted.

The following schema fragment describes the structure of the <ResourceAccessStatement> element:

```
  <xs:element name="ResourceAccessStatement"
type="sec:ResourceAccessStatementType"
substitutionGroup="saml:SubjectStatement"/>

  <xs:complexType name="ResourceAccessStatementType">
     <xs:complexContent>
        <xs:extension base="saml:SubjectStatementAbstractType">
           <xs:sequence>
              <xs:group ref="disco:ResourceIDGroup"/>
              <xs:sequence minOccurs="0">
              <!-- This is the name of the proxy and it SHOULD carry
                  SubjectConfirmation information to authorize the
                  ProxySubject to act on behalf of the
                  Subject inherited from
                  SubjectStatementAbstractType -->
                 <xs:element name="ProxySubject" type="saml:SubjectType"/>
                 <xs:element ref="sec:SessionContext" minOccurs="0"/>
              </xs:sequence>
           </xs:sequence>
        </xs:extension>
     </xs:complexContent>
  </xs:complexType>
```

The <ResourceAccessStatement> derives from <saml:SubjectStatementAbstractType> and MUST adhere to the following constraints:

- It MUST contain either a <disco:ResourceID> or a <disco:EncryptedResourceID> element which adheres to the processing rules of [LibertyDisco]
- It MAY include a <ProxySubject> element which adheres to the structure and semantics described in this specification
- It MAY include a <SessionContext> element which adheres to the structure and semantics described in this specification

8. 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.

8.1. 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 a Policy Decision Point (PDP) and potentially a coarse grained Policy Enforcement Point (PEP) to 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.

The Sender may authenticate to the Recipient using one of three general methods Section 6. Using any of these methods, both the Access Authorization PDP and the Proxy Authorization PDP may be located either at the TTP or the recipient. Although in principle all permutations of authentication mechanism, Access Authorization PDP location, and Proxy Authorization PDP location are possible.

When the PDP is located at the TTP, the TTP must issue an assertion with respect to Sender authorizations for consumption by the Service - this assertion will likely be presented to the Sender/Proxy for inclusion in the request to the Recipient. Note that if the TTP policies do not grant access to the particular resource the TTP may also act as a PEP and not issue an assertion.

When the PDP for both types of authorization decisions is located at the TTP, the two different assertions issued by the TTP logically collapse into a single authorization assertion , e.g. 'Invocation Identity can perform operation' and 'Sender can proxy for Invocation Identity' collapses to 'Sender can perform operation'. The authorization decision that the TTP creates could reflect this. It may still however be desirable for both assertions to be sent.

Authentication and authorization authorities may be co-located. When the Sender is relying on a particular TTP for both authentication (through SAML holder-of-key) and either types of authorization decision, some optimization may be possible through that TTP issuing 'combined' assertions.

8.2. Authorization Mechanism

It is RECOMMENDED that this mechanism utilize the Web Services Security SAML Profile [wss-saml] as the means by which the message sender authenticates to the recipient. Each communicating peer performs message level authentication 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 assures the consumer of the assertion that the subject confirmation key is that of the intended sender. Thus the senders subject confirmation key can be recognized by the recipient as belonging to the remote peer. 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 demonstrate its identity to the recipient and the ability for the accessing 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 resource access message, the sender identity and the invocation identity. Thus the authorization model supports a constrained proxy mechanism that permits a proxy (the sender) to access the resource on behalf of some other system entity.

8.3. Authorization Data Generation

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

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 requirements.

8.3.1. 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.

Presuming the requesting entity meets the necessary authorization criteria, the assertion issuing authority constructs the assertion in accordance with the following rules:

- The assertion MUST include at most one of the following statements; `<SessionContextStatement>` or a `<ResourceAccessStatement>`.

- The `<saml:Subject>` element MUST describe the invocation identity. The invocation identity MUST be either that of the sender or another system entity on whose behalf the sender (as a proxy) is authorized to act.

- The assertion MUST describe the invocation identity within the `<saml:Subject>` element of the base statement.

- When the invocation identity is that of the sender the `<saml:Subject>` element of the base statement type MUST be constructed as follows:
  The `<saml:Subject>` element MUST include a `<saml:SubjectConfirmation>` element with a `<saml:ConfirmationMethod>` of `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`.
  The subject confirmation key MUST be specified within the `<saml:Subject>` element by including a `<ds:KeyInfo>` element in the `<saml:SubjectConfirmation>` element.

- When the invocation identity is NOT that of the sender the `<saml:Subject>` element of the base statement type MUST be constructed as follows:
  The assertion SHOULD protect the privacy of the named entity with a `<lib:EncryptedNameIdentifier>` element.
  The authority MUST include a `<saml:SubjectConfirmation>` element in this subject and specify that the sender will vouch for the subject by indicating a `<saml:ConfirmationMethod>` of *urn:oasis:names:tc:SAML:1.0:cm:sender-vouches*.
  Additionally, the assertion MUST identify the proxy and its confirmation requirements by including a `<ProxySubject>` element.. This element MUST be constructed in accordance with the constraints defined in ProxySubject Schema (Section 7.1).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:AssertionIDReference>` which is a reference to the assertion bearing it. Also as the chain builds the assertion should be augmented with `<ProxyTransitedStatement>`. The issuer should include a `<ProxyTransitedStatement>` for each proxy that has participated in the progression of assertion issuance. See Proxy Chaining (Section 8.3.2) for a recommendation on constructing the proxy chain.

- For statements of type `<ResourceAccessStatement>` the assertion issuing authority MUST adhere to the construction constraints described in ResourceAccessStatement Schema (Section 7.6).
  The assertion issuing authority MAY describe the authentication status of the interacting party by including a `<SessionContext>` element. The `<SessionContext>` element MUST adhere to the construction constraints described in SessionContext Schema (Section 7.6).

- For statements of type `<SessionContextStatement>` the assertion issuing authority MUST adhere to the construction constraints described in SessionContextStatement Schema (Section 7.5) and SessionContext Schema (Section 7.6).

- The assertion MUST be signed by the assertion issuing authority in accordance with the signing requirements specified in [SAMLCore11].

8.3.2. Proxy Chaining

In some operational settings it may be necessary to carry the chain of proxies traversed. The following algorithm describes how an assertion issuing authority could formulate the proxy chain.

It is presumed that when a system entity interacts with the assertion issuing authority, the system entity will include a claim that contains a `<ProxyTransitedStatement>` bearing `<ProxyInfoConfirmationData>`. 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 includes the `<saml:AssertionIDReference>` of the assertion which the initial requester presented to the system entity needing subsequent proxy capability. It is presumed that the assertion issuing authority decorates assertions with `<saml:AssertionIDReference>` within the `<saml:Advice>` element for assertions which it deems to be proxiable. Thus, the assertion issuer could use this information to locate the requester's assertion and add it to the list of proxies transited. Given the assertion the proxy chain can be formulated and a new chain created by appending the most recently transited proxy to the chain. The result will be an `<saml:Assertion>` comprised of `<ProxyTransitedStatement>` elements for each of the proxies transited. It is recommended that this assertion be carried within an `<saml:Advice>` element of the assertion issued to the proxy.

The following example depicts an assertion containing advice regarding transited proxies.

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="0"
  AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
  Issuer="authority.example.com"
  IssueInstant="2004-04-01T16:58:33.173Z">

  <saml:Advice>
    <saml:AssertionIDReference>refers to this assertion</saml:AssertionIDReference>
    <saml:Assertion>
      <!-- This statement reflects the first proxy transited  -->
      <ProxyTransitedStatement>
         <Issuer>authority.example.com</Issuer>
         <IssueInstant>2004-04-01T16:58:30.173Z</IssueInstant>
         <saml:AssertionIDReference>
           <!-- refers to assertion first.example.com confirmed to
               authority.example.com to get an assertion which named
               first.example.com the ProxySubject. -->
         </saml:AssertionIDReference>
         <saml:Subject>
          <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
            http://first.example.com/
          </saml:NameIdentifier>
        </saml:Subject>
      </ProxyTransitedStatement>
```

```
      <ProxyTransitedStatement>
         <Issuer>authority.example.com</Issuer>
         <IssueInstant>2004-04-01T16:58:32.173Z</IssueInstant>
         <saml:AssertionIDReference>
          <!-- refers to assertion second.example.com confirmed to
              authority.example.com to get an assertion which named
              second.example.com the ProxySubject. -->
         </saml:AssertionIDReference>
         <saml:Subject>
          <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
           http://second.example.com/
          </saml:NameIdentifier>
        </saml:Subject>
      </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="2004-04-01T16:57:20Z"
      NotOnOrAfter="2004-04-01T21:42:43Z">
      <saml:AudienceRestrictionCondition>
          <saml:Audience>http://fourth.example.com</saml:Audience>
      </saml:AudienceRestrictionCondition>
  </saml:Conditions>

  <!-- The AuthenticationStatement carries information
      that describes the identity of the entity this assertion
      was issued to (the Subject) and the method the Subject
      authenticated to the assertion issuing authority -->
  <saml:AuthenticationStatement
      AuthenticationMethod="urn:ietf:rfc:2246"
      AuthenticationInstant="2004-04-01T16:57:30.000Z">
      <saml:Subject>
          <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
           http://third.example.com/
          </saml:NameIdentifier>
      </saml:Subject>
  </saml:AuthenticationStatement>

  <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
      <saml:Subject>
       <!-- the name identifier of the interacting entity -->
       <saml:NameIdentifier ID="#abesyd"
          Format="urn:liberty:iff:nameid:encrypted">
          HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw
          A23B45...C569UXR3==
       </saml:NameIdentifier>
       <SubjectConfirmation>
        <!-- The confirmed sender of this message vouches for the
            identity carried in this subject. -->
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
        </ConfirmationMethod>
       </SubjectConfirmation>
      </saml:Subject>
      <ProxySubject>
       <!-- the name identifier of the sender -->
       <NameIdentifier format="urn:liberty:iff:nameid:entityID">
        http://third.example.com/
       </NameIdentifier>
       <SubjectConfirmation>
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
```

```
        </ConfirmationMethod>
        <!-- This keyinfo is the key by which the sender must prove
            possession.  -->
        <ds:KeyInfo>
          <ds:KeyName>
            CN=third.example.com,OU=Services R US,O=Service Nation,...
          </ds:KeyName>
          <ds:KeyValue>...</ds:KeyValue>
        </ds:KeyInfo>
      </SubjectConfirmation>
    </ProxySubject>
  </ResourceAccessStatement>
  <!-- signature by the authority over the assertion -->
  <ds:Signature>...</ds:Signature>
</saml:Assertion>
```

8.4. Presenting Authorization Data

Interactions with identity-based web services may rely on the conveyance of authorization information. In general, a trusted authority issues the authorization data. In such a setting the authorization information would be sent along with the identity-based web service request to the recipient. See Authorization Data Generation (Section 8.3) for details as to how this data is acquired and formulated.

8.4.1. 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).

- In accordance with the above requirement the sender MUST include the resultant XML signature in a `<ds:Signature>` element. The signature element SHOULD be decorated with a ds:KeyInfo which MUST resolve to the subject confirmation key.
  To assist the receiver in processing the authorization information, the sender decorates the `<ds:Signature>` with a `<ds:KeyInfo>` which SHOULD include a `<wsse:SecurityTokenReference>` element in accordance with the [wss-sms].

8.5. 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 8.4).

The recipient determines the invocation identity by inspecting the `<saml:Subject>` element and the proxy identity by inspecting the `<ProxySubject>` if present. When the `<ProxySubject>` element is present it also describes the subject confirmation obligation of the proxy. 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.

8.5.1. Processing Rules

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

9. Identity-Based Web Service Examples (Informative)

The following are examples demonstrating the various use cases supported by this specification.

9.1. Conveyance of Sender as Invocation Identity

Recall that the authorization model describes a mechanism for an assertion issuing authority to bind the invocation identity to an assertion by specifying the invocation identity within the `<saml:Subject>`. The following example demonstrates the construction of a `<saml:Subject>` to accomplish this binding:

```
<Subject>
  <!-- the name identifier of the sender -->
  <NameIdentifier format="urn:liberty:iff:nameid:entityID">
    http://serviceprovider.com/
  </NameIdentifier>
  <SubjectConfirmation>
    <ConfirmationMethod>
      urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
    </ConfirmationMethod>
    <!-- This keyinfo is the key by which the sender must prove
      possession. Note the KeyName MAY but is NOT REQUIRED to
      match the above NameIdentifier -->
    <ds:KeyInfo>
      <ds:KeyName>
        CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
      </ds:KeyName>
      <ds:KeyValue>...</ds:KeyValue>
    </ds:KeyInfo>
  </SubjectConfirmation>
</Subject>
```

Contents in the above example worth particular mention include:

- The assertion issuing authority specifies the `<saml:ConfirmationMethod>` with the value of `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`
- The assertion issuing authority included a `<ds:KeyInfo>` element to indicate which key the `<saml:Subject>` must demonstrate possession of to meet the above confirmation obligation.

What follows is a more complete example that depicts a request to access an identity-based web service that carries authorization data to the recipient such that the sender identity and the invocation identity are the same. The resource for which the sender is attempting to access is described in a ResourceAccessStatement bound to the assertion.

The interpretation of the assertion can be expressed as follows:

According to the policies of the assertion issuing authority the sender must adhere to the confirmation requirements expressed in the encapsulated ResourceAccessStatement. To meet the obligation the message sender;

- MUST successfully demonstrate possession of the key referenced in the ResourceAccessStatement's SubjectConfirmation element which is bound to the Assertion
- AND the message receiver has a trusted path from the signing certificate to an issuing Certification Authority.

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.

```
<?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">

 <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:1.0:assertion"
    MajorVersion="1" MinorVersion="0"
    AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
    Issuer="idp.example.com"
    IssueInstant="2004-04-01T16:58:33.173Z">

    <!-- 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="2004-04-01T16:57:20Z"
        NotOnOrAfter="2004-04-01T21:42:43Z">
        <saml:AudienceRestrictionCondition>
            <saml:Audience>http://webserviceprovider.com</saml:Audience>
        </saml:AudienceRestrictionCondition>
    </saml:Conditions>

    <!-- The AuthenticationStatement carries information
        that describes the identity of the entity this assertion
        was issued to (the Subject) and the method the Subject
        authenticated to the assertion issuing authority -->
    <saml:AuthenticationStatement
       AuthenticationMethod="urn:ietf:rfc:2246"
       AuthenticationInstant="2004-04-01T16:57:30.000Z">
       <saml:Subject>
        <saml:NameIdentifier
            format="urn:liberty:iff:nameid:entityID"
            NameQualifier="http://AffiliationStation.com/">
         http://serviceprovider.com/
        </saml:NameIdentifier>
       </saml:Subject>
    </saml:AuthenticationStatement>

    <!-- The Subject of the ResourceAccessStatement specifies use of
        holder-of-key which indicates the subject of the statement
        is the sender of the message and is to be treated as the
        invocation identity for any access control decisions -->
    <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
     <saml:Subject>
      <!-- the name identifier of the sender -->
      <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
       http://serviceprovider.com/
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
       <saml:ConfirmationMethod>
        urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
```

```
        </saml:ConfirmationMethod>
        <!-- This keyinfo is the key by which the sender must prove
            possession. Note the KeyName MAY but is NOT REQUIRED to
            match the above NameIdentifier -->
        <ds:KeyInfo>
          <ds:KeyName>
            CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
          </ds:KeyName>
          <ds:KeyValue>...</ds:KeyValue>
        </ds:KeyInfo>
       </saml:SubjectConfirmation>
      </saml:Subject>

      <!-- This ResourceID describes the resource for which
        the sender is attempting to access. -->
      <disco:ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>
     </ResourceAccessStatement>
     <!-- signature by the authority over the assertion -->
     <ds:Signature>...</ds:Signature>
    </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:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
            ValueType="saml:Assertion" />
       </wsse:SecurityTokenReference>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
     </ds:SignatureValue>
    </ds:Signature>
   </wsse:Security>
  </s:Header>
  <s:Body wsu:Id="MsgBody">
     <pp:Modify>
          <!-- this is an ID-SIS-PP Modify message -->
     </pp:Modify>
  </s:Body>
</s:Envelope>
```

9.2. Conveyance of Sender as Proxy

This authorization model supports environments where the sender of a message is acting as a proxy on behalf of another system entity. The following example demonstrates the construction of a `<ProxySubject>` to accomplish this binding:

```
 <ProxySubject>
  <!-- the name identifier of the proxy -->
  <NameIdentifier format="urn:liberty:iff:nameid:entityID">
   http://serviceprovider.com/
  </NameIdentifier>
```

```
  <SubjectConfirmation>
   <ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
   </ConfirmationMethod>
   <!-- This keyinfo is the key by which the proxy must prove
    possession. Note the KeyName MAY but is NOT REQUIRED to
    match the above NameIdentifier -->
   <ds:KeyInfo>
    <ds:KeyName>
     CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
    </ds:KeyName>
    <ds:KeyValue>...</ds:KeyValue>
   </ds:KeyInfo>
  </SubjectConfirmation>
 </ProxySubject>
```

Contents in the above example worth particular mention include:

- The assertion issuing authority generated assertion indicates that the `<saml:Subject>` element is that of the system entity on whose behalf the sender is proxying.
- The assertion includes a `<ProxySubject>` element so as to convey the identity of the proxy to the relying party.
- The `<ProxySubject>` element specifies the `<saml:ConfirmationMethod>` with a value of `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`.
- The `<saml:SubjectConfirmation>` element includes a `<ds:KeyInfo>` element which indicates which key the proxy must prove possession of to meet the confirmation obligation.

The following example, is a more complete demonstration of a request invoking an identity-based web service which carries authorization data to the recipient such that the sender can act as a proxy on behalf of the entity described in the subject of the resource access statement.

```
<?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">

  <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:1.0:assertion"
     MajorVersion="1" MinorVersion="0"
     AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
     Issuer="idp.example.com"
     IssueInstant="2004-04-01T16:58:33.173Z">

     <!-- 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="2004-04-01T16:57:20Z"
         NotOnOrAfter="2004-04-01T21:42:43Z">
         <saml:AudienceRestrictionCondition>
             <saml:Audience>http://webserviceprovider.com</saml:Audience>
```

```
        </saml:AudienceRestrictionCondition>
    </saml:Conditions>

    <!-- The AuthenticationStatement carries information
        that describes the identity of the entity this assertion
        was issued to (the Subject) and the method the Subject
        authenticated to the assertion issuing authority -->
    <saml:AuthenticationStatement
       AuthenticationMethod="urn:ietf:rfc:2246"
       AuthenticationInstant="2004-04-01T16:57:30.000Z">
       <saml:Subject>
        <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
          http://serviceprovider.com/
        </saml:NameIdentifier>
       </saml:Subject>
    </saml:AuthenticationStatement>

    <!-- Subject of the ResourceAccessStatement specifies a
        confirmation method of sender-vouches. This indicates that the
        subject of the statement is being vouched for by the sender of
        the message. The sender identity is conveyed in the
        ProxySubject element. In order for the ProxySubject to vouch
        for the Subject the ProxySubject must meet the holder-of-key
        confirmation obligation. The proxy does so by demonstrating
        possession of the described key. In this scenario the Subject of
        the message is to be treated as the invocation identity (for
        any access control decisions) -->

    <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
     <saml:Subject>
      <!-- the name identifier of the interacting entity -->
      <saml:NameIdentifier ID="#abesyd"
         Format="urn:liberty:iff:nameid:encrypted">
         HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw
         A23B45...C569UXR3==
      </saml:NameIdentifier>
      <SubjectConfirmation>
       <!-- The confirmed sender of this message vouches for the
           identity carried in this subject. -->
       <ConfirmationMethod>
        urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
       </ConfirmationMethod>
      </SubjectConfirmation>
    </saml:Subject>
    <ProxySubject>
      <!-- the name identifier of the sender -->
      <NameIdentifier format="urn:liberty:iff:nameid:entityID">
       http://serviceprovider.com/
      </NameIdentifier>
      <SubjectConfirmation>
       <ConfirmationMethod>
        urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
       </ConfirmationMethod>
       <!-- This keyinfo is the key by which the sender must prove
           possession.  -->
       <ds:KeyInfo>
        <ds:KeyName>
         CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
        </ds:KeyName>
        <ds:KeyValue>...</ds:KeyValue>
       </ds:KeyInfo>
      </SubjectConfirmation>
    </ProxySubject>

    <!-- This ResourceID describes the resource for which
      the sender is attempting to access. -->
    <disco:ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>
```

```
      </ResourceAccessStatement>
      <!-- signature by the authority over the assertion -->
      <ds:Signature>...</ds:Signature>
    </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:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
    ValueType="saml:Assertion" />
       </wsse:SecurityTokenReference>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
    </ds:SignatureValue>
   </ds:Signature>
  </wsse:Security>
 </s:Header>
 <s:Body wsu:Id="MsgBody">
   <pp:Modify>
       <!-- this is an ID-SIS-PP Modify message -->
   </pp:Modify>
 </s:Body>
</s:Envelope>
```

9.3. Session Context of the Interacting Entity

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.

The following example is similar to the above example with the addition of conveying session context of the entity (principal) interacting with the message sender.

```
<?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">

  <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:1.0:assertion"
    AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
    Issuer="idp.example.com"
    IssueInstant="2004-04-01T16:58:33.173Z">

 <!-- 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="2004-04-01T16:57:20Z"
     NotOnOrAfter="2004-04-01T21:42:43Z">
     <saml:AudienceRestrictionCondition>
         <saml:Audience>http://webserviceprovider.com</saml:Audience>
     </saml:AudienceRestrictionCondition>
 </saml:Conditions>

 <!-- The AuthenticationStatement carries information
     that describes the identity of the entity this assertion
     was issued to (the Subject) and the method the Subject
     authenticated to the assertion issuing authority -->
 <saml:AuthenticationStatement
    AuthenticationMethod="urn:ietf:rfc:2246"
    AuthenticationInstant="2004-04-01T16:57:30.000Z">
    <saml:Subject>
     <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
      http://serviceprovider.com/
     </saml:NameIdentifier>
    </saml:Subject>
 </saml:AuthenticationStatement>

 <!-- subject of the ResourceAccessStatement specifies use of
     holder-of-key which indicates the subject of the statement
     is the sender of the message and is to be treated as the
     invocation identity for any access control decisions -->
 <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
  <Subject>
   <!-- the name identifier of the sender -->
   <NameIdentifier format="urn:liberty:iff:nameid:entityID">
    http://serviceprovider.com/
   </NameIdentifier>
   <SubjectConfirmation>
    <ConfirmationMethod>
     urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
    </ConfirmationMethod>
    <!-- This keyinfo is the key by which the sender must prove
        possession. Note the KeyName MAY but is NOT REQUIRED to
        match the above NameIdentifier -->
    <ds:KeyInfo>
     <ds:KeyName>
      CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
     </ds:KeyName>
     <ds:KeyValue>...</ds:KeyValue>
    </ds:KeyInfo>
   </SubjectConfirmation>
  </Subject>

  <!-- This ResourceID describes the resource for which
    the sender is attempting to access. -->
  <disco:ResourceID>http://foo.com/d0CQF8elJTDLmzEo</disco:ResourceID>
  <!-- The session context of the entity interacting with the
      request sender -->
  <SessionContext xmlns="urn:liberty:sec:2003-08"
       AuthenticationInstant="" AssertionIssueInstant="">
    <SessionSubject>
     <saml:NameIdentifier ID="#abesyd"
      Format="urn:liberty:iff:nameid:encrypted">
```

```
            HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw
            A23B45C569UkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw89XR3==
          </saml:NameIdentifier>
        </SessionSubject>
        <AuthnContext
           xmlns="urn:liberty:ac:2003-08">
           ...
        </AuthnContext>
        <ProviderID>http://serviceprovider.com/</ProviderID>
      </SessionContext>
     </ResourceAccessStatement>
     <!-- signature by the authority over the assertion -->
     <ds:Signature>...</ds:Signature>
   </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:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
    ValueType="saml:Assertion" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <ds:SignatureValue>
     HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
   </ds:SignatureValue>
  </ds:Signature>
 </wsse:Security>
  </s:Header>
  <s:Body wsu:Id="MsgBody">
    <pp:Modify>
         <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
  </s:Body>
</s:Envelope>
```

9.4. SAML Bearer Token Message Authentication

The following example demonstrates this message authentication mechanism by supplying a SAML security 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:2003-08">

 <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:1.0:assertion"
  MajorVersion="1" MinorVersion="0"
  AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
  Issuer="idp.example.com"
  IssueInstant="2004-04-01T16:58:33.173Z">

  <!-- 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="2004-04-01T16:57:20Z"
      NotOnOrAfter="2004-04-01T21:42:43Z">
      <saml:AudienceRestrictionCondition>
          <saml:Audience>http://webserviceprovider.com</saml:Audience>
      </saml:AudienceRestrictionCondition>
  </saml:Conditions>

  <!-- The AuthenticationStatement carries information
      that describes the identity of the entity this assertion
      was issued to (the Subject) and the method the Subject
      authenticated to the assertion issuing authority -->

  <saml:AuthenticationStatement
     AuthenticationMethod="urn:ietf:rfc:2246"
     AuthenticationInstant="2004-04-01T16:57:30.000Z">
     <saml:Subject>
      <saml:NameIdentifier
          format="urn:liberty:iff:nameid:entityID"
          NameQualifier="http://AffiliationStation.com/">
        http://serviceprovider.com/
      </saml:NameIdentifier>
     </saml:Subject>
  </saml:AuthenticationStatement>

  <!-- The Subject of the ResourceAccessStatement specifies use of
      bearer which indicates the subject of the statement
      is the sender of the message and is to be treated as the
      invocation identity for any access control decisions -->
  <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
   <saml:Subject>
    <!-- the name identifier of the sender -->
    <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
     http://serviceprovider.com/
    </saml:NameIdentifier>
    <saml:SubjectConfirmation>
     <saml:ConfirmationMethod>
      urn:oasis:names:tc:SAML:1.0:cm:bearer
     </saml:ConfirmationMethod>
    </saml:SubjectConfirmation>
   </saml:Subject>

   <!-- This ResourceID describes the resource for which
     the sender is attempting to access. -->
   <disco:ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>
  </ResourceAccessStatement>
  <!-- signature by the authority over the assertion -->
  <ds:Signature>...</ds:Signature>
 </saml:Assertion>
 <!-- this is the reference to the bearer token -->
 <wsse:SecurityTokenReference>
     <wsse:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
         ValueType="saml:Assertion" />
 </wsse:SecurityTokenReference>
</wsse:Security>
```

```
  </s:Header>
  <s:Body wsu:Id="MsgBody">
     <pp:Modify>
          <!-- this is an ID-SIS-PP Modify message -->
     </pp:Modify>
  </s:Body>
</s:Envelope>
```

The next 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:2003-08">

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

    <wsse:Security>
     <!-- Custom binary security token -->
     <wsse:BinarySecurityToken
         ValueType="anyNSPrefix:ServiceSessionContext"
         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-sec
urity-1.0#Base64Binary"
         wsu:Id="bst" >
     mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
     YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpM+sF2BnpND118f/mXCv3XbWhiL
     xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
     QXdp1/HIYQo5tIvCAM4pGk8nJFh6JrLsOEnT887aJRaasvBAAQ27C7D4Dmpt01aC
     FqLEQ98/lt6nkFmf7oiuZkID++xQXn74LWOvdNlki43VaSXWcQAjzCzirHSuVX1N
     QvAsufa9Vghnry5Blxe2VzwitMDwiRCS/bpbRQAFEbQmR2FyeSBGLiBFbGxpc29u
     IDxnYXJ5LmVsbGlzb25Ac3VuLmNvbT6JARUDBRA0Z5icfpHfi79/fM0BARwaB/sG
     YHj+fpvMgRZev/i0DyZX+s6YyMZKeJ4pVHeboFP7KaP0R+VvAP0qojK+6ITUyX2w
     R3eqeJPMbWqmOA/EAYkYE/xcqrq2ddSq2SG43530/TTOfY+ENXttltVhBdJ79KLx
     8fR2f9jLKJqQBu2MRKpy5EdJ1qmthKQm/SGTKRz8uncs5BtmJxkAbskuSi6Ys24E
     Pv0r97dW/uTfh7VM8+SA/hkCF6QVE1UzvgpKwEpoh2DZiuzvwAFqV/tINZRHGhCg
     TNLvyz+5yYXSAY3nr8UPzNJ9QUXrsmzBGDSlpqp3GO7kL0VHN//B/5GLSVcofzpA
     xj/JP+41N4sDJGkyCWwqiQEeBBABAgAJBQI+d0xwAhkBAAoJEPCJEJL9ultFpMgH
     9AzI8pmuPKxv3dQcuqZ+rJRsy2YYuuSkWpj97n5PFWvBGTSAu2+2wo3uLn8A596w
     n4MVShtx5SC2rMKKZABJ8ObqtbbS1tQaIJmPg471qmnHjazeqbPfPwpQHzQ66cje
     De/3QbxBD/rPXV2SiyECed0qRsbuC9Oo3TonrJBOp6+Hs6jSkjGvQeJjvutuklMN
     A9TOd0CKN1RiEUWl4zwef7cmHWjWyfC64l8pqMFLC7XrYE7pXAL2Y6pi8Ta5njGL
     1dWryWzSDMCEunOt5wiuUYqZ+BXvy11kp2iKmi56ioTg5UHxGJqr6oZONDwMDIhW
     sI9v1kuHhJuWz8DZiZO1i7QgR2FyeSBFbGxpc29uIDxnZmVAaW50ZXJoYWNrLm5l
     dD6JARQDBRA+d1WR8IkQkv26W0UBAXgsB/UROD8wayj9v7gMK3K9Idxk/3K16myl
     m0Q5mzFkXoLZ6EJ3wZlpxteR9oeTo2F/5tJ0k9SFNaeIfFuipVGz9y+iDHHVKyQw
     kDGg7YB5+fK1siebpUnIemvhmngrUzLnmbOJDpBy+UukRGjRLhDsuEXN8fpGb27d
     ddo2odK31nR9OpRPGo/F2mkduatD28MMPVn4RpOKw8Nx7PIIxVPnTXGgfLY2PDOO
     Dk5he7KszA3rJul9Dof0Ii9nLHlOXiHwXWFx7le66vwlHCIaNwpvU8BXSeIgbKDA
     ZzFMfUHsKyTdMo9l+ByDk/jLsGsvZ61tROShVWSwO0rC8pKa3sVmSMy0C2dmZUBz
     dW4uY29tiQETAwUQP3plwvCJEJL9ultFAQGRDgfwmhqrrlACqYAr2a2yFoex0gIz
     NrTQvMjRWw5EyzoGu9KMQ5ilsBIpIHCcA6LY/Y6rb0qsrP7Pu0Z082uuQAlfpRzs
     i4lHsZDOeKKAiw7G3bJO+fDpkwYPHC7YFObof45Y71BWO+OBfKrMb73ZfgYYGKIc
     tECofkVO3fvNHNEeDIEzhvY2o783JOGbdN34P5NcLre69eLPF3KNhonLQMVxlNmh
     0kwl5rUckRPAPy4WgKv/VQEZtXSPmx9t4x3jUjc+yDtSdvTnBMwEHUU3/Pn8TICa
     XsvFX/55u0POntxFoi1A+0UpsCGrGpdzv1q7tRmFsF5aOP1Um79Qg1O/5060Gkdh
     cnkgRWxsaXNvbiA8Z2ZlQHN1bi5jb20+iQEUAwUQP3pmAvCJEJL9ultFAQF1twf0
```

```
     CAY7B8Nb74w+mYYyHS+UXCrPQR21vs5DjzuKooX7j6pJHDQqhfss24NLBvvpufZa
     uTE27fDIx+HC0SK5cjGUTqoX/4nkMe+HM87vPcChbS3lTGT+yxVjyiQ9BIei5mX2
     QTl9RkS3ZDXNux32uONDRX7dykNX6fYkKRGserWHhdXlHppmmvLodKCK/sZkkqzf
     VT4r9ytfpXBluelOV93X8RUz4ecZcDm9e+IEG+pQjnvgrSgac1NrW5K/CJEOUUjh
     oGTrym0Ziutezhrw/gOeLVtkywsMgDr77gWZxRvw01w1ogtUdTceuRBIDANj+KVZ
     vLKlTCaGAUNIjkiDDgti
     =OuKj
    </wsse:BinarySecurityToken>

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

10. XSD

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

<xs:schema targetNamespace="urn:liberty:sec:2003-08"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
        xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
        xmlns:ac="urn:liberty:ac:2003-08"
        xmlns:lib="urn:liberty:iff:2003-08"
        xmlns:disco="urn:liberty:disco:2003-08"
        xmlns:md="urn:liberty:metadata:2003-08"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:sec="urn:liberty:sec:2003-08"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
   <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
           schemaLocation="oasis-sstc-saml-schema-assertion-1.1.xsd"/>
   <xs:import namespace="urn:liberty:iff:2003-08"
           schemaLocation="liberty-idff-protocols-schema-1.2-errata-v3.0.xsd"/>
   <xs:import namespace="urn:liberty:disco:2003-08"
           schemaLocation="liberty-idwsf-disco-svc-v1.2.xsd"/>
   <xs:import namespace="urn:liberty:ac:2003-08"
           schemaLocation="liberty-authentication-context-v1.3.xsd"/>
   <xs:import namespace="urn:liberty:metadata:2003-08"
           schemaLocation="liberty-metadata-v1.1.xsd"/>
   <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"
           schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
           schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-s
chema.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 1.2
14th December 2004

Copyright (c) 2003-2005 Liberty Alliance participants, see
http://www.projectliberty.org/specs/idwsf_1_1_copyrights.php

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

   <xs:element name="ValidityRestrictionCondition" type="sec:ValidityRestrictionConditionType"/>
   <xs:complexType name="ValidityRestrictionConditionType">
      <xs:complexContent>
         <xs:extension base="saml:ConditionAbstractType">
           <xs:sequence>
             <xs:element name="NumberOfUses" type="xs:integer"/>
           </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="ProxySubject" substitutionGroup="saml:Subject"
            type="saml:SubjectType"/>
   <xs:annotation>
      <xs:documentation>ProxyTransitedStatement is a
      SubjectStatement which MAY carry specific subject confirmation
      data </xs:documentation>
   </xs:annotation>
   <xs:element name="ProxyTransitedStatement"
 type="saml:SubjectStatementAbstractType"/>

```

```
  <xs:annotation>
     <xs:documentation>
      ProxyInfoConfirmationData may be relied upon to
      corroborate the path information carried in a
      ProxyTransitedStatement
</xs:documentation>
  </xs:annotation>
  <xs:element name="ProxyInfoConfirmationDat a"
          type="sec:ProxyInfoConfirmationType"/>
  <xs:complexType name="ProxyInfoConfirmationType">
    <xs:sequence>
      <xs:element ref="saml:AssertionIDReference"/>
      <xs:element name="Issuer" type="xs:string"/>
      <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:complexType>

  <xs:element name="SessionContext" type="sec:SessionContextType"/>
  <xs:complexType name="SessionContextType">
     <xs:sequence>
        <xs:element name="SessionSubject" type="lib:SubjectType"/>
        <xs:element name="ProviderID" type="md:entityIDType"/>
        <xs:element ref="lib:RequestAuthnContext" minOccurs="0" maxOccurs="1"/>
        <!-- The system entity for which this context applies
    is privacy protect by the SessionSubject -->
     </xs:sequence>
     <xs:attribute name="SessionIndex" type="xs:string" use="optional"/>
     <xs:attribute name="AuthenticationInstant" type="xs:dateTime" use="required"/>
     <xs:attribute name="AssertionIssueInstant" type="xs:dateTime" use="required"/>
  </xs:complexType>

  <xs:element name="SessionContextStatement"
          type="sec:SessionContextStatementType"
          substitutionGroup="saml:SubjectStatement"/>

  <xs:complexType name="SessionContextStatementType">
     <xs:complexContent>
        <xs:extension base="saml:SubjectStatementAbstractType">
           <xs:sequence>
              <!-- This is the name of the proxy and it SHOULD carry
                  SubjectConfirmation information to authorize the
                  ProxySubject to act on behalf of the
                  Subject inherited from
                  SubjectStatementAbstractType -->
              <xs:element name="ProxySubject"
                       type="saml:SubjectType"  minOccurs="0"/>
              <xs:element ref="sec:SessionContext"/>
           </xs:sequence>
        </xs:extension>
     </xs:complexContent>
  </xs:complexType>

  <xs:element name="ResourceAccessStatement"
type="sec:ResourceAccessStatementType"
substitutionGroup="saml:SubjectStatement"/>

  <xs:complexType name="ResourceAccessStatementType">
     <xs:complexContent>
        <xs:extension base="saml:SubjectStatementAbst ractType">
           <xs:sequence>
              <xs:group ref="disco:ResourceIDGroup"/>
              <xs:sequence minOccurs="0">
              <!-- This is the name of the proxy and it SHOULD carry
                  SubjectConfirmation information to authorize the
                  ProxySubject to act on behalf of the
```

```
                    Subject inherited from
                    SubjectStatementAbstractType -->
                   <xs:element name="ProxySubject" type="saml:SubjectType"/>
                   <xs:element ref="sec:SessionContext" minOccurs="0"/>
                </xs:sequence>
             </xs:sequence>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>

</xs:schema>
```

Bibliography

Normative

[LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 1.3, Liberty Alliance Project (14 December 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-v3.0, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyDisco] Sergent, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 1.2, Liberty Alliance Project (12 December 2004). *http://www.projectliberty.org/specs*

[LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. " Liberty ID-WSF SOAP Binding Specification ," Version 1.2, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyGlossary] "Liberty Technical Glossary," Version 1.4, Liberty Alliance Project (14 Dec 2004). *http://www.projectliberty.org/specs* Hodges, Jeff, eds.

[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*

[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*

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

[wss-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/*