Liberty ID-WSF Security Profiles

Version: 1.0-08

Editors:
Gary Ellison, Sun Microsystems, Inc.

Contributors:
John Linn, RSA Laboratories
Frederick Hirsch, Nokia
Robert Aarts, Nokia
Paul Madsen, Entrust
Scott Cantor, The Ohio State University/Internet2

Notice

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

Revision History

Revision: 08 **Date:** 11 Apr 2003

Post SAN. Cleaned up requirements section. Closed bugs: 145-147, 152-157, 159-163.

Revision: 07 **Date:** 05 Apr 2003

Consolidation and factoring of authentication mechanisms and authorization. Abandoned use of Attribute statements. Created three top level SAML statements to support the primitives of authentication and authorization described in the Discovery service.

Revision: 06 **Date:** 26 Mar 2003

No substantive changes intended. mostly clarifications around self consistency and accuracy. Also started to tease apart authn for authz

Revision: 06 **Date:** 21 Mar 2003

nits Add AuthnContext to schema fragment.

Revision: 05 **Date:** 17 Mar 2003

Incorporate feedback from IAH and on the list

Revision: 04 **Date:** 24 Feb 2003

10 hours befor IAH, More feedback from Frederick Hirsch.

Revision: 03 **Date:** 20 Feb 2003

Incorporate feedback from the list. Restructure per John's remarks

Revision: 02 **Date:** 04 Feb 2003

post MIA

Revision: 01 **Date:** 08 Jan 2003

initial, post PHX

Contents

1. Abstract

This document specifies security protocol profiles for securing the consumption of identity services. An identity 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 service. The specified mechanisms provide for authentication, signing and encryption operations. XML-Signature and XML-Encryption are utilized to provide the associated transformations and processing semantics to accommodate the message authentication and protection functionality. OASIS WS-Security compliant header elements communicate the relevant security information, i.e., a SAML assertion, along with the protected message.

2. Overview of Identity Services Authorization (Informative)

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

An identity 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 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 services may be accessed by system entities. The access may be direct or with the assistance of an active intermediary. To access an identity service a system entity must interact with a specific service instance which exposes some resource.

Given the above description, we strongly believe that access control policies must be enforced by identity services. The authorization decision to access an identity 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 service, thus the identity 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 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 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 service. Of course this does not preclude the identity service from making additional policy decisions based on other criteria.

Our definition of an identity 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 service must consider the status of the resource owner for a given request to access a resource.

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

3. Notation and Terminology

This section specifies the notations, namespaces and 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 Section 3(at the end of this document).

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
idwsfsec	urn:liberty:id-wsf:sec:1.0
invc	urn:liberty:isf:invocation-invc:1.0"
ac	urn:liberty:ac:1.2
lib	urn:liberty:iff:1.2
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://schemas.xmlsoap.org/ws/2002/xx/secext
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 [LibertyGloss].

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

153 • Sender – the initial SOAP sender. A sender is a proxy when its identity differs from the invocation identity.

154 • Proxy – entity whose authenticated identity, according to the recipient, differs from that of the entity making the
155 invocation.

156 • Trusted Authority – a Trusted Third Party (TTP) which issues and vouches for SAML assertions

157 • Invocation Identity – party invoking a service.

158 • Service – invocation responder, providing a service. Ultimate message processor.

4. Security Requirements

This section details the security requirements which this specification must support. This section first presents an use case scenarios envisioned for identity services. We then followup 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, proxy rights for sender to make request on behalf of invocation identity
- Authentication of the response
- Authentication of service identity
- Authentication context and session status of the interacting entity
- Authorization of invocation identity to access service or resource

Note that the authorization framework 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 the invocation framework to support a controlled non-transitive proxy capability.

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 rules apply to all profiles in this specification, unless otherwise noted by the individual profile.

1. Messages may need to be kept confidential and inhibit unauthorized disclosure, either when in transit or when stored persistently. Different mechanisms will be required depending on the situation.
2. Messages need to arrive at the intended recipient without undetected changes, whether accidental or as a result of unauthorized tampering. 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.
3. 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. This is important to prevent man-in-the-middle attacks.
4. 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.
5. For signing and verification of protocol messages, communicating entities should use signing keys that are distinct from confidentiality keys.
6. The privacy requirements of the participants must be maintained

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 typical 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 certificates based authentication over a SSL/TLS 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 sender.

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 (non-transitive) proxy capability whereby a consumer of an identity service (the intermediate system entity or proxy) can act on behalf of another system entity (the subject) to access an identity 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 a token 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 an authority playing the roles of Policy Administration Point and Policy Decision Point, the recipient must still implement some degree of policy decisions and enforcement.

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 intermediaries in the message path) then transport layer protection mechanisms may suffice to ensure the integrity and confidentiality of the message exchange. However, this mechanism may not fully address the privacy and confidentiality requirements of information supplied by a trusted authority. For example the authorization data my contain sensitive information. To accommodate this requirement the trusted authority and ultimate recipient MUST rely upon the mechanisms specified in Encrypted Identifiers and Encrypted URI SHOULD be used.

1. Messages between sender and recipient MUST have their integrity protected and confidentiality MUST be ensured. This requirement MUST be met with suitable SSL 3.0 or TLS 1.0 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

2. 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.

5.2. Message Layer 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 [WSScore] to encrypt the content 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 my contain sensitive information. To accommodate this requirement the trusted authority and ultimate recipient MUST rely upon the mechanisms specified in Encrypted Identifiers and Encrypted URI SHOULD be used.

5.3. Message Layer Identifier Privacy Protection

Under certain usage scenarios the information conveyed by the Trusted Authority for consumption by the identity service may contain privacy sensitive data. However, this data generally passes through the system entity accessing the particular identity service. One example is the name identifier from the federated namespace of the authority and the identity service. Another sensitive data item may be the URI which has some association with the identity service and the principal on whose behalf the sender is acting.

5.3.1. Encrypted Identifiers

The invocation identity conveyed in the `<saml:Subject>` MUST be resolvable in the namespace of the consuming service instance. This is problematic given the existing privacy requirement to hinder collusion and correlation by not sharing identities across service providers. To continue to meet this requirement [LIBbind] defines `<lib:EncryptedSubject>` element which extends `<saml:Subject>`.

The `<xenc:EncryptedKey>` element MUST exhibit nonce-like semantics. Otherwise it would circumvent the privacy requirement for which the mechanism is intended to address. The `<xenc:EncryptedKey>` element is used for key transport. Therefore, the `xenc:Algorithm` attribute of the `<xenc:EncryptionMethod>` element MUST be one of the URIs designated for key transport as defined in [XMLEnc].

5.3.2. Encrypted URI

At times it may also be necessary to privacy protect the contents of a URI so as to not release sensitive information to an intermediary. To accomplish this we first define a wrapper element for a URI and then an encrypted variant of the element.

A resource is described with the `<idwsfsec:ResourceIdentifier>` element.

Figure 1.

```
   <xs:element name="ResourceIdentifier" type="ResourceIdentifierType"/>
   <xs:complexType name="ResourceIdentifierType">
     <xs:simpleContent>
 <xs:extension base="xs:anyURI">
  <xs:attribute name="id" type="xs:ID"/>
 </xs:extension>
    </xs:simpleContent>
   </xs:complexType>

   <xs:element name="EncryptedResourceIdentifier"
     type="EncryptedResourceIdentifierType"/>
   <xs:complexType name="EncryptedResourceIdentifierType">
     <xs:sequence>
 <xs:element ref="xenc:EncryptedData"/>
 <xs:element ref="xenc:EncryptedKey" minOccurs="0"/>
    </xs:sequence>
```

```
322     </xs:complexType>
323
```

6. Peer Authentication Mechanisms

The mechanism profiles described here specify the forms of authentication technology which can be used to authenticate communicating peers. The multiplicity of mechanisms specified is necessary to accommodate the optimal deployment scenarios. That said with each mechanism described the deployment setting will be noted as guidance only.

6.1. Transport Layer Sender Authentication

The mechanism MAY be used in the absence of active intermediaries in the message path. Its primary function is to provide for the authentication of the communicating peers and to leverage confidentiality and integrity features of the SSL 3.0 or TLS 1.0 or compatible versions. This profile MAY be used in combination with other security mechanisms in this specification.

Authentication of both the sender and the recipient is REQUIRED.

Messages between sender and recipient MUST have their integrity protected and confidentiality MUST be ensured

The sender and recipient MUST implement the following authentication methods:

1. SSL 3.0 or TLS 1.0 mutual authentication with X.509 client and server-side certificates
2. ID-FF 1.2 AuthnResponse over TLS

If a sender uses SSL 3.0 or TLS 1.0, it MUST use a X.509 client-side certificate.

If a recipient uses SSL 3.0 or TLS 1.0, it MUST use a X.509 server-side certificate.

6.2. Message Layer Sender Authentication Mechanisms

In the presence of active intermediaries the sender MUST rely upon message layer authentication mechanisms rather than the mechanisms specified in Transport Layer Sender Authentication. Therefore, the sender MUST authenticate at the messaging layer by using either the X.509 Certificate Message Layer Sender Authentication or the SAML Assertion Message Layer Sender Authentication authentication mechanism as specified below.

To aid in processing messages authenticated using the following mechanisms, security tokens MUST qualify their usage by specifying the `<wsse:Usage>` attribute with a QName value of `idwsfsec:MessageAuthentication`.

6.2.1. X.509 Certificate Message Layer Sender Authentication

This mechanism utilizes the Web Services Security X.509 Profile as the means by which the message sender authenticates to the recipient.

The sender performs message layer authentication by demonstrating proof of possession of a key which is recognized by the recipient as belonging to the sender. It is RECOMMENDED that a sender distribute the signing certificate in-band. The sender presents the security token to the recipient by inserting it in the `<wsse:Security>` and demonstrating knowledge of the key by signing elements of the message. That is, possession of the key MUST be demonstrated to bind the message to the sender. The sender MUST sign over the contents of the `<invc:request>` and the `<S:Body>` elements.

Figure 2.

```
<wsse:Security xmlns:wsse="...">
  <wsse:BinarySecurityToken
```

```
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
        Id="X509Token" ValueType="wsse:X509v3"
        EncodingType="wsse:Base64Binary">
        MIIEZzCCA9CgAwIBAgIQEmtJZc0...
      </wsse:BinarySecurityToken>
      ...

     <ds:Signature>
      <ds:SignedInfo>
        ...
  <ds:Reference URI="#IsfHeader">
        <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 Usage="idwsfsec:MessageAuthentication">
        <wsse:Reference URI="#X509Token"/>
      </wsse:SecurityTokenReference>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
     </ds:SignatureValue>
    </ds:Signature>
   </wsse:Security>
```

6.2.1.1. Processing Rules

The processing rules specified in the [WSScore] and [WSSx509] MUST be followed.

6.2.2. SAML Assertion Message Layer Sender Authentication

This mechanism utilizes the Web Services Security SAML Profile as the means by which the message sender authenticates to the recipient.

The sender performs message layer authentication by demonstrating proof of possession of a key which is recognized by the recipient as belonging to the sender. For this scenario it is RECOMMENDED that a trusted authority issue an assertion that binds the sender to its key. The sender presents this assertion (a security token) to the recipient by placing the assertion in the `<wsse:Security>` and demonstrating knowledge of the key by signing elements of the message. That is, possession of the key MUST be demonstrated to bind the message to the sender. The sender MUST sign over the contents of the `<invc:request>` and the `<S:Body>` elements.

7. Message Layer Authorization Model

The Message Layer Authorization Model specifies OPTIONAL mechanisms to convey authorization and resource access information (supplied by a trusted third party) which 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 which the subject's certificate accomplishes for authentication purposes.

The following table depicts the various security mechanisms and their usage.

Table 2. Authentication and Authorization

Sender Role	Remote Policy	Local Policy	AuthN Mech.	AuthZ Mech.	Confidentiality/ Integrity
Self	none	PDP/PEP	SSL Client	local	SSL
Self	none	PDP/PEP	X.509 Message	local	XML Enc./XML Sig.
Self	PIP	PDP/PEP	SSL Client	SAML holder-of-key	SSL
Self	PIP	PDP/PEP	X.509 Message	SAML holder-of-key	XML Enc./XML Sig.
Self	PIP	PDP/PEP	SAML Message	SAML holder-of-key	XML Enc./XML Sig.
Proxy	PIP	PDP/PEP	SSL Client	SAML sender-vouches	SSL
Proxy	PIP	PDP/PEP	X.509 Message	SAML sender-vouches	XML Enc./XML Sig.
Proxy	PIP	PDP/PEP	SAML Message	SAML sender-vouches	XML Enc./XML Sig.

The authorization model supports the issueance of assertions which 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 even the ability for the accessing entity to access the resource on behalf of some other system entity. This latter facility suggest 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 restricted (non-transitive) proxy capability which permits a proxy (the sender) to access the resource on behalf of some other system entity.

To aid in processing of the authorization information `<wsse:SecurityTokenReference>` elements MUST qualify their usage by specifying the `<wsse:Usage>` attribute with a QName value of `idwsfsec:MessageAuthorization`.

7.1. Resource Access Statement

Resource access information is captured in a `<idwsfsec:ResourceAccessStatement>` element.

Figure 3.

```
<xs:element name="ResourceAccessStatement"
  type="ResourceAccessStatementType"/>
<xs:complexType name="ResourceAccessStatementType">
   <xs:complexContent>
      <xs:extension base="saml:SubjectStatementAbstractType">
        <xs:sequence>
<xs:choice>
            <xs:element ref="ResourceIdentifier"/>
            <xs:element ref="EncryptedResourceIdentifier"/>
</xs:choice>
          <!- This is the name of the proxy and it MUST carry
            SubjectConfirmation info to authz the
```

```
                ProxySubject to act on behalf of the
                Subject inherited from SubjectStatementAbstractType ->
   <xs:sequence minOccurs="0">
      <xs:element name="ProxySubject" type="saml:SubjectType"/>
      <xs:element ref="SessionContext" minOccurs="0"/>
   </xs:sequence>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
```

7.2. 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 which conveys the session status of the interacting entity. The sender would present this data to the recipient.

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

Figure 4.

```
   <xs:element name="SessionContext" type="SessionContextType"/>
   <xs:complexType name="SessionContextType">
     <xs:sequence>
<!- The system entity for which this context applies
    is privacy protect by the EncryptedSubject ->
<xs:element name="EncryptedSubject" type="lib:EncryptedSubjectType"/>
<xs:element name="ProviderID" type="xs:anyURI"/>
<xs:element ref="lib:AuthnContext"/>
    </xs:sequence>
    <xs:attribute name="AuthenticationInstant" type="xs:dateTime"
 use="required"/>
    <xs:attribute name="SessionEstablishmentInstant"
 type="xs:dateTime"
 use="required"/>
  </xs:complexType>

  <xs:element name="SessionContextStatement"
    type="SessionContextStatementType"/>
  <xs:complexType name="SessionContextStatementType">
     <xs:complexContent>
        <xs:extension base="saml:SubjectStatementAbstractType">
          <xs:sequence>
             <xs:element ref="SessionContext"/>
          </xs:sequence>
        </xs:extension>
     </xs:complexContent>
  </xs:complexType>
```

The SessionContext element applies to the `<lib:EncryptedSubject>` carried in the extended `<saml:SubjectStatementType>`.

7.3. Conveyance of Sender as Invocation Identity

The authorization model relies upon an assertion issuing authority to bind the invocation identity to an assertion by specifying the invocation identity within the `<saml:Subject>` of the resource access statements of the assertion. An example of the `<saml:Subject>` follows:

Figure 5.

```
<Subject>
  <!- the name identifier of the sender ->
  <NameIdentifier format="#X509SubjectName">
    CN=serviceprovider.com OU=Services R US, O=Service Nation,...
  </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>
```

Additionally, the assertion issuing authority specifies the `<saml:ConfirmationMethod>` within the `<saml:SubjectConfirmation>` element thus dictating the method by which the sender must prove to the recipient that the sender is the entity described in the `<saml:Subject>` of the assertion. A `<saml:ConfirmationMethod>` with the value of `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key` MUST be specified when the sender identity is to be considered for authorization decisions.

When the `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key` `<saml:ConfirmationMethod>` is specified then the `<saml:SubjectConfirmation>` element MUST include a `<ds:KeyInfo>` element which indicates what key the `<saml:Subject>` MUST prove possession of to meet the confirmation obligation.

7.4. Conveyance of Sender as Proxy

For scenarios where the sender of a message is acting as a proxy the issuing authority MUST generate an assertion where the `<saml:Subject>` element is that of a system entity on whose behalf the sender is proxying.

However to convey the proxy identity, the issuing authority MUST generate the assertion such that it includes a `<saml:ProxySubject>` element. The `<saml:ProxySubject>` element MUST also provide the `<saml:ConfirmationMethod>` with a value of `urn:oasis:names:tc:SAML:1.0:cm:sender-vouches`. Indicating `sender-vouches` informs to the recipient the role of the proxy. The `<saml:SubjectConfirmation>` element MUST include a `<ds:KeyInfo>` element which indicates what key the proxy must prove possession of to meet the confirmation obligation. The following example demonstrates the construction of a `<idwsfsec:ProxySubject>`:

Figure 6.

```
<ProxySubject>
  <!- the name identifier of the proxy ->
  <NameIdentifier format="#X509SubjectName">
    CN=serviceprovider.com OU=Services R US, O=Service Nation,...
  </NameIdentifier>
  <SubjectConfirmation>
    <ConfirmationMethod>
      urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
    </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>
```

The recipient can always determine the invocation identity by inspecting the `<saml:Subject>` element and the proxy identity by inspecting the `<idwsfsec:ProxySubject>` which also describes the subject confirmation method the proxy MUST demonstrate. In a proxy situation the `<saml:ConfirmationMethod>` within the `<saml:SubjectConfirmation>` element MUST have a value of `urn:oasis:names:tc:SAML:1.0:cm:sender-vouches`. This indicates that the identity in the `<saml:Subject>` element is not the sender. As always the recipient SHOULD use the invocation identity to make its authorization decisions. However the recipient SHOULD also determine whether it permits the `<ProxySubject>` to access the resource on behalf of the `<saml:Subject>`.

For the `urn:oasis:names:tc:SAML:1.0:cm:sender-vouches` `<saml:ConfirmationMethod>` the `<saml:SubjectConfirmation>` element MUST convey additional information to assist the recipient in determining the trustworthiness of the message and the sender by including `<ds:KeyInfo>`. The recipient MUST verify that the message was secured using the specified key.

8. Generating Authorization Data

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

In addition to managing the registration and discovery of identity services the Trusted Authority may act as a centralized policy decision point whereby it issues assertions regarding authorization decisions made based on the access control policies enforced for a given identity service, resource and the identity of the sender. The makeup of this assertion MUST reflect the information necessary to accommodate the authorization requirements specified at the time the service was registered.

8.1. Trusted Authority Assertion Generation Processing Rules

The following processing rules describe the steps the assertion issuing authority MUST take to generate an assertion to distribute to the sender for subsequent service or resource access request messages.

- The trusted authority MUST enforce any access control policies pertaining to the resource which the sender is attempting to locate.
- If according to the trusted authority's policies the sender is NOT permitted to access the resource a failure indication SHALL be returned.
- The trusted authority MUST construct a `<saml:Assertion>` which conveys the resource access information and any supporting authorization data.
- The assertion MUST include at most one of the following statements; `<idwsfsec:SessionContextStatement>` or a `<idwsfsec:ResourceAccessStatement>`. In addition to the statement specific elements the statements SHOULD bear some or all of the following elements:

  - 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 trusted authority MUST make this selection contingent upon its access control policies and any policies being enforced by the trusted authority on behalf of the system entity offering the resource.
  - When the identity in the `<saml:Subject>` represents the sender the trusted authority MUST include a `<saml:SubjectConfirmation>` element with a `<saml:ConfirmationMethod>` of `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`. The issuing authority MUST specify the subject confirmation key by including a `<ds:KeyInfo>` element.
    When the invocation identity is not that of the sender, the trusted authority SHOULD describe the invocation identity with a `<saml:Subject>` element of type `<saml:EncryptedSubject>` and the authority SHOULD NOT include a `<saml:SubjectConfirmation>` element. To convey the identity of the proxy the trusted authority MUST include a `<idwsfsec:ProxySubject>` element. The `<idwsfsec:ProxySubject>` MUST include a `<saml:SubjectConfirmation>` element with a `<saml:ConfirmationMethod>` of `urn:oasis:names:tc:SAML:1.0:cm:sender-vouches`. The issuing authority MUST specify the subject confirmation key by inserting a `<ds:KeyInfo>` element.

- For statements of type `<idwsfsec:ResourceAccessStatement>` the trusted authority MUST describe the resource for which the assertion is bound to by including a `<idwsfsec:ResourceIdentifier>`. If the policies of the trusted authority mandate that the value of the resource not be revealed to the issuee then the Trusted Authority MUST identify the resource with an `<idwsfsec:EncryptedResourceIdentifier>` instead Encrypted URI.
  The trusted authority MAY describe the authentication status of the interacting party. by including a `<idwsfsec:SessionContext>` element.

- For statements of type `<idwsfsec:SessionContextStatement>` the trusted authority MUST describe the authentication status of the interacting party. by including a `<idwsfsec:SessionContext>` element.

- The assertion MUST be signed by the Trusted Authority in accordance to the signature requirements specified in [SAMLCore].

9. Presenting Authorization Data

Interactions with identity services may rely on authorization information to be conveyed which has been issued from a trusted authority. In such a setting the authorization information would be sent along with the identity service request to the recipient. See Generating Authorization Data for details as to how this data is acquired and formulated.

9.1. Request Sender Processing Rules

- The sender MUST authenticate to the recipient using one of the mechanisms described in Peer Authentication Mechanisms.
- The sender MUST place the issued `<saml:Assertion>` as a child of the `<wsse:Security>` element. The `<saml:Assertion>` MUST conform to the structure and content described in Generating Authorization Data. Both the `<wsse:Security>` element and the `<invc:request>` MUST be child elements of the same `<S:Header>`.
- The sender MUST demonstrate possession of the key bound by the `<saml:Assertion>` being presented by performing a signature operation. The sender MUST produce an XML Signature covering the following nodes:
  - `<invc:request>`
  - `<S:Body>`

  This operation binds the knowledge of the key to the request and the body of the message such that a cut-n-paste attack is not possible. The properties of isf request processing MUST prevent replay attacks
- The sender MUST sign the requisite elements with the key corresponding to that which is described in the `<saml:SubjectConfirmation>` element of the issued assertion.
- The sender must include the resultant XML signature in a `<ds:Signature>` element in within the `<wsse:Security>` element. The `<ds:Signature>` element MUST include a `<wsse:SecurityTokenReference>` element in accordance with the [WSScore]. The `<wsse:SecurityTokenReference>` element MUST be qualified with usage information by specifying the `<wsse:Usage>` attribute. The value MUST be a QName value of `idwsfsec:MessageAuthentication` or `idwsfsec:MessageAuthorization` depending on what action the sender attempting.

10. 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 section Presenting Authorization Data.

10.1. Request Recipient Processing Rules

- The recipient MUST authenticate the sender. See Peer Authentication Mechanisms.
- The recipient MUST locate the `<saml:Assertion>` (security token) and verify that the `<saml:Assertion>` is structured in accordance with Presenting Authorization Data
- The recipient MUST locate the `<ds:KeyInfo>` element bound to the `<saml:SubjectConfirmation>` element of the `<saml:Assertion>` and determine that it represents the same key used to sign the message. That is, the recipient MUST corroborate that the bound subject confirmation key matches the key used to sign the message.
- The recipient MUST verify the `<ds:Signature>` element carried inside the `<wsse:Security>` header block. This validation MUST conform to the core validation as described in the [XMLSig]. The recipient SHOULD validate the certificate and verify the certificate revocation status, as appropriate to the risk of incorrect authentication.
- The recipient MUST determine that it trusts the key used to sign the message.
- The recipient MUST verify that this signature covers the requisite portions of the message.
    - `<invc:request>`
    - `<S:Body>`
- The recipient MUST validate the signature of the `<saml:Assertion>`. The recipient SHOULD validate the certificate and verify the certificate revocation status, as appropriate to the risk of incorrect authentication.
- The recipient MUST determine that it trusts the authority which signed the `<saml:Assertion>`.

Following the above processing the recipient should enforce access control policies based in the verified information. Possible policy decisions follow:

- The recipient SHOULD corroborate the `idwsfsec:ResourceIdentifier` within the `<saml:ResourceAccessStatement>` implies the resource being accessed.

11. Identity Service Examples (Informative)

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

11.1. Sender as Invocation Identity Example

The following example depicts a request to access an identity service which carries authorization data to the recipient such that the sender identity and the invocation identity are the same. The resource offering for which the sender is attempting to invoke is described in an attribute carried within the same attribute statement bound to the assertion.

The interpretation of the assertion can be expressed as follows:

In accordance with the discovery policies of the service described in the encapsulated ResourceOffering which is bound to an enclosed Attribute Statement, the message sender may invoke the service hosted by the Audience;

- IF AND ONLY IF the message sender can successfully demonstrate possession of the key bound to the SubjectConfirmation element of 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.

Figure 7.

```
<S:Header>
  <invc:request  wsu:Id="IsfHeader">
    ...
  </invc:request>

  <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="2002-06-19T16:58:33.173Z">
    <!- 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:oasis:names:tc:SAML:1.0:assertion">
     <Subject>
       <!- the name identifier of the sender ->
       <NameIdentifier format="#X509SubjectName">
        CN=serviceprovider.com OU=Services R US, O=Service Nation,...
       </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 matche 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 ResourceIdentifier describes the resouce for which
        the sender is attempting to access. ->
      <ResourceIdentifier>http://example.com/disco/d0CQF8elJTDLmzEo</ResourceIdentifier>
      <!- 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="#IsfHeader">
          <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 Usage="idwsfsec:MessageAuthentication">
         <saml:AssertionIDReference>2sxJu9g/vvLG9sAN9bKp/8q0NKU=
         </saml:AssertionIDReference>
       </wsse:SecurityTokenReference>
       <wsse:SecurityTokenReference Usage="idwsfsec:MessageAuthorization">
         <saml:AssertionIDReference>2sxJu9g/vvLG9sAN9bKp/8q0NKU=
         </saml:AssertionIDReference>
       </wsse:SecurityTokenReference>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
    </ds:SignatureValue>
   </ds:Signature>
  </wsse:Security>
 </S:Header>
 <S:Body wsu:Id="MsgBody">
 </S:Body>
```

11.2. Session Context Example

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.

Figure 8.

```
<S:Header>
  <invc:request  wsu:Id="IsfHeader">
    ...
  </invc:request>

  <wsse:Security>
    <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
        AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
        Issuer="idp.example.com"
        IssueInstant="2002-06-19T16:58:33.173Z">
      <!- 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:oasis:names:tc:SAML:1.0:assertion">
        <Subject>
          <!- the name identifier of the sender ->
          <NameIdentifier format="#X509SubjectName">
            CN=serviceprovider.com OU=Services R US, O=Service Nation,...
          </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 matche 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 ResourceIdentifier describes the resouce for which
        the sender is attempting to access. ->
      <ResourceIdentifier>http://foo.com/d0CQF8elJTDLmzEo</ResourceIdentifier>
      <!- The session context of the entity interacting with the
          request sender ->
      <SessionContext xmlns="urn:liberty:id-wsf:sec:1.0"
            AuthenticationInstant="" SessionEstablishmentInstant="">
        <lib:EncryptedSubject>...</lib:EncryptedSubject>
        <AuthnContext
           xmlns="http://www.projectliberty.org/schemas/authctx/2002/05">
           ...
        </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="#IsfHeader">
         <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 Usage="idwsfsec:MessageAuthentication">
         <saml:AssertionIDReference>2sxJu9g/vvLG9sAN9bKp/8q0NKU=
         </saml:AssertionIDReference>
       </wsse:SecurityTokenReference>
       <wsse:SecurityTokenReference Usage="idwsfsec:MessageAuthorization">
         <saml:AssertionIDReference>2sxJu9g/vvLG9sAN9bKp/8q0NKU=
         </saml:AssertionIDReference>
       </wsse:SecurityTokenReference>
     </ds:KeyInfo>
     <ds:SignatureValue>
      HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
    </ds:SignatureValue>
   </ds:Signature>
  </wsse:Security>
 </S:Header>
  <S:Body wsu:Id="MsgBody">
  </S:Body>
```

11.3. Sender Acting as Proxy Example

The following example demonstrates a request invoking an identity 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 attribute statement.

Figure 9.

```
<invc:request  wsu:Id="IsfHeader">
  ...
</invc:request>

  <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="2002-06-19T16:58:33.173Z">
    <!- 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:oasis:names:tc:SAML:1.0:assertion">
      <lib:EncryptedSubject>
        <NameIdentifier></NameIdentifier>
        <!- the name identifier of the interacting entity ->
        <EncryptedNameIdentifier ID="#abesyd"
      Type="http://www.w3.org/2001/04/xmlenc#Content">
      <CipherData>
   <CipherValue>A23B45C569UXR3==</CipherValue>
      </CipherData>

        </EncryptedNameIdentifier>
        <EncryptedKey Id='NIEK'>
          <EncryptionMethod Algorithm="..."/>
       <ds:KeyInfo>
  <ds:KeyName>Recipient</ds:KeyName>
       </ds:KeyInfo>
       <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
        <ReferenceList>
   <DataReference URI="#abesyd"/>
            </ReferenceList>
        </EncryptedKey>
      </lib:EncryptedSubject>
      <ProxySubject>
       <!- the name identifier of the sender ->
       <NameIdentifier format="#X509SubjectName">
         CN=serviceprovider.com OU=Services R US, O=Service Nation,...
       </NameIdentifier>
       <SubjectConfirmation>
        <ConfirmationMethod>
         urn:oasis:names:tc:SAML:1.0:cm:sendervouches
        </ConfirmationMethod>
        <!- This keyinfo is the key by which the sender must prove
         possession. Note the KeyName MAY matche 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>

      <!- This ResourceIdentifier describes the resouce for which
        the sender is attempting to access. ->
      <ResourceIdentifier>http://example.com/disco/d0CQF8elJTDLmzEo</ResourceIdentifier>
      </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="#IsfHeader">
        <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 Usage="idwsfsec:MessageAuthentication">
       <saml:AssertionIDReference>2sxJu9g/vvLG9sAN9bKp/8q0NKU=
       </saml:AssertionIDReference>
     </wsse:SecurityTokenReference>
     <wsse:SecurityTokenReference Usage="idwsfsec:MessageAuthorization">
       <saml:AssertionIDReference>2sxJu9g/vvLG9sAN9bKp/8q0NKU=
       </saml:AssertionIDReference>
     </wsse:SecurityTokenReference>
   </ds:KeyInfo>
   <ds:SignatureValue>
    HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
  </ds:SignatureValue>
 </ds:Signature>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="MsgBody">
</S:Body>
```

12. XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!- $Id: lib-arch-security-fmwk.xsd,v 1.4 2003/04/13 19:59:13 gellison Exp $ ->
<!- Author: Gary Ellison ->
<!- Last editor: $Author: gellison $ ->
<!- $Revision: 1.4 $ ->
<xs:schema targetNamespace="urn:liberty:id-wsf:sec:1.0"
   xmlns="urn:liberty:id-wsf:sec:1.0"
   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
   xmlns:lib="urn:liberty:iff:1.2"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-
open.org/committees/security/docs/cs-sstc-schema-assertion-01.xsd"/>
   <xs:import namespace="urn:liberty:iff:1.2" schemaLocation="lib-arch-protocols-schemas.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:annotation>
      <xs:documentation>Editor: Gary Ellison</xs:documentation>
      <xs:documentation>Copyright 2003 Liberty Alliance Project</xs:documentation>
   </xs:annotation>


   <xs:element name="MessageAuthentication" type="xs:QName"/>
   <xs:element name="RequesterAuthorization" type="xs:QName"/>

   <xs:element name="ValidityRestrictionCondition"
 type="ValidityRestrictionConditionType"/>
   <xs:complexType name="ValidityRestrictionConditionType">
     <xs:sequence>
       <xs:element name="NumberOfUses" type="xs:integer"/>
     </xs:sequence>
   </xs:complexType>

   <xs:element name="SessionContext" type="SessionContextType"/>
   <xs:complexType name="SessionContextType">
     <xs:sequence>
 <!- The system entity for which this context applies
     is privacy protect by the EncryptedSubject ->
 <xs:element name="EncryptedSubject" type="lib:EncryptedSubjectType"/>
 <xs:element name="ProviderID" type="xs:anyURI"/>
 <xs:element ref="lib:AuthnContext"/>
     </xs:sequence>
     <xs:attribute name="AuthenticationInstant" type="xs:dateTime"
  use="required"/>
     <xs:attribute name="SessionEstablishmentInstant"
  type="xs:dateTime"
  use="required"/>
   </xs:complexType>

   <xs:element name="SessionContextStatement"
     type="SessionContextStatementType"/>
   <xs:complexType name="SessionContextStatementType">
      <xs:complexContent>
         <xs:extension base="saml:SubjectStatementAbstractType">
           <xs:sequence>
              <xs:element ref="SessionContext"/>
           </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>


   <xs:element name="ResourceIdentifier" type="ResourceIdentifierType"/>
   <xs:complexType name="ResourceIdentifierType">
     <xs:simpleContent>
 <xs:extension base="xs:anyURI">
   <xs:attribute name="id" type="xs:ID"/>
 </xs:extension>
     </xs:simpleContent>
   </xs:complexType>

   <xs:element name="EncryptedResourceIdentifier"
```

```
      type="EncryptedResourceIdentifierType"/>
    <xs:complexType name="EncryptedResourceIdentifierType">
      <xs:sequence>
  <xs:element ref="xenc:EncryptedData"/>
  <xs:element ref="xenc:EncryptedKey" minOccurs="0"/>
     </xs:sequence>
    </xs:complexType>

    <xs:element name="ResourceAccessStatement"
      type="ResourceAccessStatementType"/>
    <xs:complexType name="ResourceAccessStatementType">
       <xs:complexContent>
          <xs:extension base="saml:SubjectStatementAbstractType">
            <xs:sequence>
    <xs:choice>
                <xs:element ref="ResourceIdentifier"/>
                <xs:element ref="EncryptedResourceIdentifier"/>
    </xs:choice>
              <!- This is the name of the proxy and it MUST carry
                 SubjectConfirmation info to authz the
                 ProxySubject to act on behalf of the
                 Subject inherited from SubjectStatementAbstractType ->
    <xs:sequence minOccurs="0">
       <xs:element name="ProxySubject" type="saml:SubjectType"/>
       <xs:element ref="SessionContext" minOccurs="0"/>
    </xs:sequence>
           </xs:sequence>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

1062 # Bibliography

Normative Specifications

[LIBac] Kemp, J., Madsen, P., eds. (2003). Liberty Alliance Project> "Liberty Authentication Context Specification," Version 1.2,

[LIBbind] Kemp, J., Cantor, S., eds. (2003). Liberty Alliance Project> "Liberty ID-FF Protocols and Schema Specification," Version 1.2,

[SAMLCore] Hallam-Baker, P., Maler, E., Eds., , eds. (2002). OASIS "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)," http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf,

[WSSsaml] Hallam-Baker, P., Kaler, C., Monzillo, R., Nadalin, A., eds. (2002). OASIS "Web Services Security: SAML Token Profile," WSS-SAML-06.pdf,

[WSSx509] Hallam-Baker, P., Kaler, C., Monzillo, R., Nadalin, A., eds. (2002). OASIS "Web Services Security: X509 Certificate Token Profile," WSS-X509-03.pdf,

[WSSkrb] Hallam-Baker, P., Kaler, C., Monzillo, R., Nadalin, A., eds. (2002). OASIS "Web Services Security: Kerberos Token Profile," WSS-Kerberos-03.pdf,

[WSScore] Hallam-Baker, P., Kaler, C., Monzillo, R., Nadalin, A., eds. (2002). OASIS "Web Services Security: SOAP Message Security," WSS-SOAPMessageSecurity-11-0303-merged.pdf,

[XMLSig] (2001). W3C "XML Signature Syntax and Processing,"

[XMLEnc] (2002). W3C "XML Encryption Syntax and Processing,"

[RFC3268] Chown, P., eds. (2002). The Internet Society "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)," http://www.ietf.org/rfc/rfc3268.txt,