



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

1

Notice

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

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

15 Copyright © 2004-2005 ADAE; Adobe Systems; America Online, Inc.; American Express Company; Avatier
16 Corporation; Axalto; Bank of America Corporation; BIPAC; Computer Associates International, Inc.; DataPower
17 Technology, Inc.; Diversinet Corp.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.; Ericsson; Fidelity Investments;
18 Forum Systems, Inc. ; France Telecom; Gamefederation; Gemplus; General Motors; Giesecke & Devrient GmbH;
19 Hewlett-Packard Company; IBM Corporation; Intel Corporation; Intuit Inc.; Kantega; Kayak Interactive; MasterCard
20 International; Mobile Telephone Networks (Pty) Ltd; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nippon
21 Telegraph and Telephone Corporation; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle
22 Corporation; Ping Identity Corporation; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp
23 Laboratories of America; Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Telefonica Moviles, S.A.;
24 Trusted Network Technologies.; Trustgenix; UTI; VeriSign, Inc.; Vodafone Group Plc. All rights reserved.

25 Liberty Alliance Project
26 Licensing Administrator
27 c/o IEEE-ISTO
28 445 Hoes Lane
29 Piscataway, NJ 08855-1331, USA
30 info@projectliberty.org

31 Contents

32	1. Abstract	4
33	2. Overview of Identity-Based Web Services Authorization (Informative)	5
34	3. Notation and Terminology	6
35	4. Security Requirements (Informative)	8
36	5. Message Confidentiality and Privacy Mechanisms	11
37	6. Authentication Mechanisms	13
38	7. Supporting Schema	23
39	8. Message Authorization Model	27
40	9. Identity-Based Web Service Examples (Informative)	33
41	10. XSD	44
42	Bibliography	47

43 1. Abstract

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

53 **2. Overview of Identity-Based Web Services Authorization (Infor-** 54 **mative)**

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

56 An identity-based web service is a particular type of a web service that acts upon some resource to either retrieve
57 information about an identity, update information related to an identity, or to perform some action for the benefit of
58 some identity. A resource is either data related to some identity or a service acting for the benefit of some identity.

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

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

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

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

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

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

85 3. Notation and Terminology

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

89 3.1. Notational Conventions

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

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

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

97 3.2. Namespace

98 The following namespaces are referred to in this document:

99 **Table 1. Namespaces**

Prefix	Namespace
sec	<i>urn:liberty:sec:2003-08</i>
sb	<i>urn:liberty:sb:2003-08</i>
disco	<i>urn:liberty:disco:2003-08</i>
ac	<i>urn:liberty:ac:2003-08</i>
lib	<i>urn:liberty:iff:2003-08</i>
md	<i>urn:liberty:metadata:2003-08</i>
saml	<i>urn:oasis:names:tc:SAML:1.0:assertion</i>
S	http://www.w3.org/2002/12/soap-envelope
ds	http://www.w3.org/2000/09/xmlsig#
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

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

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

104 3.3. Terminology

105 Definitions for Liberty-specific terms can be found in [\[LibertyGlossary\]](#).

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

- 107 • Recipient – entity which receives a message that is the ultimate processor of the message

- 108 • Sender – the initial SOAP sender. A sender is a proxy when its identity differs from the invocation identity.
- 109 • Proxy – entity whose authenticated identity, according to the recipient, differs from that of the entity making the
110 invocation.
- 111 • Trusted Authority – a Trusted Third Party (TTP) that issues, and vouches for, SAML assertions
- 112 • Invocation Identity – party invoking a service.
- 113 • Service – invocation responder, providing a service. Ultimate message processor.

114 4. Security Requirements (Informative)

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

118 4.1. Security Requirements Overview

119 There are multiple facets this security specification considers:

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

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

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

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

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

- 137 • Request Authentication
- 138 • Response Authentication
- 139 • Request/Response Correlation
- 140 • Replay Protection
- 141 • Integrity Protection
- 142 • Confidentiality Protection
- 143 • Privacy Protections
- 144 • Resource Access Authorization
- 145 • Proxy Authorization
- 146 • Mitigation of denial of service attack risks

147 **4.2. Common Requirements**

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

- 149 • Messages may need to be kept confidential and inhibit unauthorized disclosure, either when in transit or when
150 stored persistently. Confidentiality may apply to the entire message, selected headers, payload, or XML portions
151 depending on application requirements.
- 152 • Messages need to arrive at the intended recipient with data integrity. SOAP intermediaries may be authorized to
153 make changes, but no unauthorized changes should be possible without detection. Integrity requirements may
154 apply to the entire message, selected headers, payload, or XML portions depending on application requirements.
- 155 • The authentication of a message sender and/or initial sender may be required by a receiver to process the message.
156 Likewise, a sender may require authentication of the response.
- 157 • Message responses must correspond to message requests and attempts to replay requests or responses should be
158 detected. Likewise the attempt to substitute requests or responses should be detected. Transaction integrity requires
159 that messages be timely and related to each other.
- 160 • The privacy requirements of the participants with respect to how their information is shared or correlated must be
161 ensured.

162 **4.3. Peer Authentication Requirements**

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

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

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

178 **4.4. Message Correlation Requirements**

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

181 **4.5. Privacy Requirements**

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

186 **4.6. Service Availability**

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

189 **4.7. Resource Access Authorization Requirements**

190 Previously we mentioned the notion of conveying both a *sender identity* and an *invocation identity*. In doing so the
191 framework accommodates a restricted proxy capability whereby a consumer of an identity-based web service (the
192 intermediate system entity or proxy) can act on behalf of another system entity (the subject) to access an identity-
193 based web service (the recipient.) To be granted the right to proxy for a subject, the intermediate system entity may
194 need to interact with a trusted authority. Based on the authority's access control policies, the authority may generate
195 and distribute an assertion authorizing the intermediary to act on behalf of the subject to the recipient. This protocol
196 framework can only convey authoritative information regarding the identities communicated to other system entities.
197 Even with the involvement of a trusted authority that makes authorization decisions permitting the proxy to access a
198 web service, the recipient should still implement a policy enforcement point.

199 **5. Message Confidentiality and Privacy Mechanisms**

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

203 **5.1. Transport Layer Channel Protection**

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

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

211 • TLS_RSA_WITH_RC4_128_SHA

212 • TLS_RSA_WITH_3DES_EDE_CBC_SHA

213 • TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

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

218 • TLS_RSA_WITH_AES_CBC_SHA

219 • TLS_DHE_DSS_WITH_AES_CBC_SHA

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

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

224 **5.2. Message Confidentiality Protection**

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

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

233 **5.3. Identifier Privacy Protection**

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

239 **5.3.1. Encrypted Name Identifiers**

240 The identity conveyed in the subject **MUST** be resolvable in the namespace of the consuming service in-
241 stance. However, this requirement is in conflict with the need to protect the privacy of the identifier when the
242 message passes through intermediaries. To accomplish this securely the `<saml:Subject>` **MUST** contain a
243 `<saml:NameIdentifier>` following the processing rules in the NameIdentifier Encryption Profile specified in
244 [[LibertyBindProf](#)].

245 **5.3.2. Encrypted URI**

246 At times it may be necessary to privacy protect the contents of a URI to deter the release of sensitive in-
247 formation to an intermediary. The [[LibertyDisco](#)] specification defines an encrypted form of a URI with the
248 `<disco:EncryptedResourceID>` schema element. This specification relies upon the semantics defined in
249 [[LibertyDisco](#)] to fulfill this privacy requirement. Thus the processing rules defined by [[LibertyDisco](#)] for the
250 `<disco:EncryptedResourceID>` element **MUST** be followed.

251 6. Authentication Mechanisms

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

- 255 • Peer Entity Authentication
- 256 • Message Authentication

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

261 The following table summarizes the authentication mechanism identifiers and their security properties. Each URI is
 262 of the form *urn:liberty:security:date:peer mechanism:message mechanism*.

263 **Table 2. Authentication Mechanisms**

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

264 6.1. Authentication Mechanism Overview (Informative)

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

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

273 The peer entity authentication mechanisms defined by this specification leverage the authentication features supplied
 274 by SSL 3.0 [SSL] or TLS 1.0 [RFC2246]. The mechanism identifier describes whether the recipient ("TLS") is
 275 unilaterally authenticated or whether each communicating peer ("ClientTLS") is mutually authenticated to the other
 276 peer. The peer entity authentication mechanisms (Section 6.2) are best suited for direct message exchanges between
 277 end systems and when the message exchange may be sufficiently trusted to not require additional attestation of the
 278 message payload. However this does not obviate the processing of subject confirmation obligations but rather enables

279 alternative and potentially optimized processing rules. Such optimizations are a matter of security policy as it applies
280 to the trust model in place between communicating entities.

281 The message authentication mechanisms indicate which attestation profile is utilized to ensure the authenticity of a
282 message. These message authentication facilities aid the deployer in the presence of intermediaries. The X.509 v3
283 Certificate mechanism (Section 6.3.1) is suited for message exchanges that generally rely upon message authentication
284 as the principle factor in making authorization decisions. The SAML Assertion mechanism (Section 6.3.2) is suited
285 for message exchanges that generally rely upon message authentication as well as the conveyance and attestation
286 of authorization information. The *Bearer* mechanism (Section 6.3.3) is based on the presence of a *bearer token* in
287 the security header of a message. In this case, the bearer token is verified for authenticity rather than proving the
288 authenticity of the message. Each operational setting has its own security and trust requirements and in some settings
289 the issuance of bearer tokens by a security token service, such as [LibertyDisco] may greatly simplify the sender's
290 processing obligations. For example, when the Discovery service indicates that a bearer mechanism is supported
291 and issues a bearer token, the sender can simply populate the security header with the tokens and send the request.
292 However this does not necessarily obviate the requirement for the recipient to process and verify the bearer token.
293 Such an optimization is a matter of security policy as it applies to the trust model in place between the communicating
294 entities.

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

307 6.2. Peer Entity Authentication

308 The Peer entity authentication mechanisms prescribed by this specification rely upon the inherent security properties
309 of the TLS/SSL protocol (sometimes referred to as transport-level security). The mechanisms described below have
310 distinct security properties regarding which peers in a message exchange are authenticated. For the mechanisms
311 that include both peer entity authentication and message authentication, optimizations regarding attestation MAY be
312 employed. For example, in environments where the signature attesting to the authenticity of the message need not
313 be retained, then it may be sufficient to rely upon the security properties of peer entity authentication to assure the
314 integrity and authenticity of the message payload.

315 6.2.1. Unilateral Peer Entity Authentication

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

- 317 • *urn:liberty:security:2003-08:TLS:null*
- 318 • *urn:liberty:security:2005-02:TLS:X509*
- 319 • *urn:liberty:security:2005-02:TLS:SAML*
- 320 • *urn:liberty:security:2005-02:TLS:Bearer*

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

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

325 **6.2.1.1. Processing Rules**

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

328 • The sender MUST authenticate the recipient.

329 • The recipient MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its
330 certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.

331 **6.2.2. Mutual Peer Entity Authentication**

332 The following URIs support mutual (sender and recipient) peer entity authentication:

333 • *urn:liberty:security:2003-08:ClientTLS:null*

334 • *urn:liberty:security:2005-02:ClientTLS:X509*

335 • *urn:liberty:security:2005-02:ClientTLS:SAML*

336 • *urn:liberty:security:2005-02:ClientTLS:Bearer*

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

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

341 **6.2.2.1. Processing Rules**

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

344 • The sender MUST authenticate the recipient AND the recipient MUST authenticate the sender.

345 • The recipient MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its
346 certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.

347 • The sender MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its
348 certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.

349 **6.3. Message Authentication**

350 The message authentication mechanisms prescribed by this specification rely upon the integrity properties imbued by
351 the application and verification of digital signatures over elements of the message payload. The mechanisms described
352 below have distinct security properties regarding authenticity of a given message. For the mechanisms that include
353 both peer entity authentication and message authentication, optimizations regarding attestation MAY be employed.
354 For example, in environments where the signature attesting to the authenticity of the message need not be retained,
355 then it may be sufficient to rely upon the properties of peer entity authentication to assure the integrity and authenticity
356 of the message payload. Another example might be when unilateral peer entity authentication of the recipient can
357 be combined with message authentication of the sender such that the response could be deemed trustworthy in the
358 absence of a signature over the payload of said response.

359 **6.3.1. X.509 v3 Certificate Message Authentication**

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

361 • *urn:liberty:security:2005-02:null:X509*

362 • *urn:liberty:security:2005-02:TLS:X509*

363 • *urn:liberty:security:2005-02:ClientTLS:X509*

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

371 For deployment settings that require message authentication independent of peer entity authentication, then the sending
372 peer MUST perform message authentication by demonstrating proof of possession of a subject confirmation key. This
373 key MUST be recognized by the recipient as belonging to the sending peer.

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

379 **6.3.1.1. Sender Processing Rules**

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

382 • The sender MUST demonstrate possession a subject confirmation key.
383 For deployment settings which REQUIRE independent message authentication, the obligation MUST be accom-
384 plished by signing elements of the message and decorating the `<wsse:Security>` element with the signature.
385 For deployment settings which DO NOT REQUIRE independent message authentication then the sender MUST
386 accomplish this obligation by decorating the security header with a `<ds:KeyInfo>` element bearing the security
387 token. This MUST be unambiguously verified to be the same certificate and key used in establishing peer entity
388 authentication. This is necessary to mitigate the threat of a certificate substitution attack. Also note that this
389 optimization only applies to the *urn:liberty:security:2005-02:ClientTLS:X509* mechanism.

- 390 • If peer entity authentication is not in use and the message is bound with [\[LibertySOAPBinding\]](#) the sender MUST
391 sign:
- 392 • The `<sb:Correlation>` header block element.
- 393 • All other header block elements that require the aforementioned security properties in accordance with the
394 security requirements prescribed in their respective specification.
- 395 • All sub-elements of the `<S:Body>`.
- 396 • If the message is signed then the sender MUST include the resultant XML signature in a `<ds:Signature>`
397 element as a child of the `<wsse:Security>` header.
398 The `<ds:Signature>` element MUST refer to the subject confirmation key with a `<ds:KeyInfo>` element which
399 SHOULD carrying a `<wsse:SecurityTokenReference>` element.

400 6.3.1.2. Recipient Processing Rules

- 401 • The recipient MUST locate the `<wsse:Security>` element for which it is the target. This MUST adhere to the
402 syntax and processing rules specified in [\[wss-sms\]](#) and [\[wss-x509\]](#).
- 403 • If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the
404 recipient MUST locate the `<ds:KeyInfo>` element bearing a security token. This token MUST be unambiguously
405 verified to be referring to the same certificate and key used in establishing peer entity authentication.
- 406 • If the message has been signed then the recipient MUST locate the `<ds:Signature>` element carried inside the
407 `<wsse:Security>` header.
408 The recipient MUST resolve the contents of the `<ds:KeyInfo>` element carried within the `<ds:Signature>`
409 and use the key it describes for validating the signed elements.
410 This validation MUST conform to the core validation rules described in [\[XMLDsig\]](#). Additionally, the recipient
411 MUST determine that it trusts the key used to sign the message, and the recipient SHOULD validate the sender's
412 certificate, verifying the certificate revocation status as appropriate to the risk of incorrect authentication.
- 413 • If peer entity authentication is not in use and the message is bound with [\[LibertySOAPBinding\]](#) the recipient
414 MUST verify the signature covers the following elements:
- 415 • The `<sb:Correlation>` header block element.
- 416 • All other header block elements that require the aforementioned security properties in accordance with the
417 security requirements prescribed in their respective specification.
- 418 • All sub-elements of the `<S:Body>`.

419 **6.3.1.3. Example X.509 v3 Message Authentication (Informative)**

420 The following example demonstrates this message authentication mechanism.

```
421 <?xml version="1.0" encoding="UTF-8"?>
422 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
423           xmlns:sb="urn:liberty:sb:2003-08"
424           xmlns:pp="urn:liberty:id-sis-pp:2003-08"
425           xmlns:sec="urn:liberty:sec:2003-08"
426           xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ws
427 security-secext-1.0.xsd">
428
429
430   <s:Header>
431     <sb:Correlation s:mustUnderstand="1"
432                   id="A13454...245"
433                   actor="http://schemas.../next"
434                   messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
435                   timestamp="2112-03-15T11:12:12Z"/>
436     <wsse:Security xmlns:wsse="...">
437       <wsse:BinarySecurityToken
438         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x
439 509-token-profile-1.0#X509v3"
440         wsu:Id="X509Token"
441         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
442 soap-message-security-1.0#Base64Binary">
443         MIIB9zCCAWSgAwIBAgIQ...
444       </wsse:BinarySecurityToken>
445       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
446         <ds:SignedInfo>
447
448           <!-- bind the correlation header -->
449           <ds:Reference URI="#A13454...245">
450             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
451             <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
452           </ds:Reference>
453           <!-- bind the security token (thwart cert substitution attacks) -->
454           <ds:Reference URI="#X509Token">
455             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
456             <ds:DigestValue>Ru4cAfeBABE...</ds:DigestValue>
457           </ds:Reference>
458           <!-- bind the body of the message -->
459           <ds:Reference URI="#MsgBody">
460             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
461             <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
462           </ds:Reference>
463         </ds:SignedInfo>
464         <ds:KeyInfo>
465           <wsse:SecurityTokenReference>
466             <wsse:Reference URI="#X509Token" />
467           </wsse:SecurityTokenReference>
468         </ds:KeyInfo>
469         <ds:SignatureValue>
470           HJJWbvqW9E84vJVQkjLLA6nNvBX7mY00TzhwBdFNDElgs SXZ5Ekw==
471         </ds:SignatureValue>
472       </ds:Signature>
473     </wsse:Security>
474   </s:Header>
475   <s:Body wsu:Id="MsgBody">
476     <pp:Modify>
477       <!-- this is an ID-SIS-PP Modify message -->
478     </pp:Modify>
479   </s:Body>
480 </s:Envelope>
481
482
```

483 **6.3.2. SAML Assertion Message Authentication**

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

485 • *urn:liberty:security:2005-02:null:SAML*

486 • *urn:liberty:security:2005-02:TLS:SAML*

487 • *urn:liberty:security:2005-02:ClientTLS:SAML*

488 These mechanisms utilize the Web Services Security SAML Token Profile [[wss-saml](#)] as the means by which the mes-
489 sage sender authenticates to the recipient. In general these mechanisms assume that a TTP issues an assertion which in-
490 cludes an `<saml:AuthenticationStatement>` and other statements derived from `<saml:SubjectStatement>`.
491 The `<saml:AuthenticationStatement>` describes the authentication event at the issuing authority (a TTP.) For
492 each of the `<saml:SubjectStatement>` bound into the assertion, the authority dictates the subject confirmation
493 obligations the subject must affirm to a relying party in order for the statement to be considered trustworthy.

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

499 These message authentication mechanisms are unilateral. That is only, the author of the message is authenticated. It
500 is not in the scope of this specification to suggest when response messages should be authenticated, but it is worth
501 noting that the mechanisms defined in [Section 6.3.1](#) could be relied upon to authenticate any response message as
502 well. Deployers should recognize, however, that independent authentication of response messages does not provide
503 the same message stream protection semantics as a mutual peer entity authentication mechanism would offer.

504 For deployment settings which require message authentication independent of peer entity authentication, then the
505 sending peer **MUST** perform message authentication by demonstrating proof of possession of a subject confirmation
506 key. This key **MUST** be recognized by the recipient as belonging to the sender.

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

512 **6.3.2.1. Sender Processing Rules**

513 • The construction and decoration of the `<wsse:Security>` header element **MUST** adhere to the rules specified in
514 the [[wss-sms](#)] and [[wss-saml](#)].

515 • The sender **MUST** present the `<saml:Assertion>` (as security token) by inserting it as a child of
516 `<wsse:Security>`.

- 517 • The sender MUST adhere to its subject confirmation obligation in accordance with the semantics of the confirma-
518 tion method described by each `<saml:SubjectStatement>` bound into the `<saml:Assertion>`.
519 For deployment settings which REQUIRE independent message authentication, the obligation MUST be accom-
520 plished by signing elements of the message and decorating the `<wsse:Security>` element with the signature.
521 For deployment settings which DO NOT REQUIRE independent message authentication then the subject confirma-
522 tion obligation may be accomplished by correlating the certificate and key used to affect peer entity authentication
523 with the certificate and key described by the subject confirmation element. To accommodate this the assertion
524 issuing authority MUST construct the assertion such that the confirmation key can be unambiguously verified to
525 be the same certificate and key used in establishing peer entity authentication. This is necessary to mitigate the
526 threat of a certificate substitution attack. It is RECOMMENDED that the certificate or certificate chain be bound
527 to the subject confirmation key.
- 528 • If peer entity authentication is not in use and the message is bound with [\[LibertySOAPBinding\]](#) the sender MUST
529 sign:
- 530 • The `<sb:Correlation>` header block element.
- 531 • All other header block elements that require the aforementioned security properties in accordance with the security
532 requirements prescribed in their respective specification.
- 533 • All sub-elements of the `<S:Body>`.
- 534 • If the message is signed the sender MUST include the resultant XML signature in a `<ds:Signature>` element as
535 a child of the `<wsse:Security>` header
536 The `<ds:Signature>` element MUST refer to the subject confirmation key with a `<ds:KeyInfo>` element. This
537 element SHOULD include a `<wsse:SecurityTokenReference>` element so that the subject confirmation key
538 can be located within the `<wsse:Security>` header. The inclusion of the reference SHOULD adhere to the
539 guidance specified in section 3.3.2 of [\[wss-saml\]](#).

540 6.3.2.2. Recipient Processing Rules

- 541 • The recipient MUST locate the `<wsse:Security>` element for which it is the target. This MUST adhere to the
542 rules specified in [\[wss-sms\]](#) and [\[wss-saml\]](#).
- 543 • The recipient MUST locate the `<saml:Assertion>` (security token) and the recipient MUST determine that it
544 trusts the authority which issued the `<saml:Assertion>`.
545 The recipient MUST validate the signature of the `<saml:Assertion>`. The recipient SHOULD validate the trust
546 semantics of the signing key, as appropriate to the risk of incorrect authentication.
- 547 • The recipient SHOULD verify the confirmation obligation of each `<saml:SubjectStatement>` bound within
548 the assertion. Only the statements which comply with the confirmation obligation bound into the statement should
549 be considered trustworthy.
- 550 • If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the
551 recipient MUST locate the `<ds:KeyInfo>` element within `<saml:SubjectConfirmation>` element. This key
552 MUST be unambiguously verified to be referring to the same certificate and key used in establishing peer entity
553 authentication.

- 554 • If the message has been signed then the recipient MUST locate the `<ds:Signature>` element carried inside the
555 `<wsse:Security>` header.
556 The recipient MUST resolve the contents of the `<ds:KeyInfo>` element carried within the `<ds:Signature>`
557 and use the key it describes for validating the signed elements.
558 This validation MUST conform to the core validation rules described in [\[XMLDsig\]](#).
559 The recipient MUST determine that it trusts the key used to sign the message. The recipient SHOULD validate the
560 sender's certificate and verify the certificate revocation status, as appropriate to the risk of incorrect authentication.
- 561 • If peer entity authentication is not in use and the message is bound with [\[LibertySOAPBinding\]](#) the recipient
562 MUST verify the signature covers the following elements:
- 563 • The `<sb:Correlation>` header block element.
- 564 • All other header block elements that require the aforementioned security properties in accordance with the security
565 requirements prescribed in their respective specification.
- 566 • All sub-elements of the `<S:Body>`.

567 **6.3.3. Bearer Token Authentication**

568 The following URIs indicate bearer mechanisms:

- 569 • *urn:liberty:security:2005-02:null:Bearer*
- 570 • *urn:liberty:security:2005-02:TLS:Bearer*
- 571 • *urn:liberty:security:2005-02:ClientTLS:Bearer*

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

575 For example, a bearer token with a `wsse:ValueType` attribute of `http://docs.oasis-open.org/wss/oasis-`
576 `wss-saml-token-profile-1.0#SAMLAssertionID` [\[wss-saml\]](#) could contain statements describing other partic-
577 ipants to a transaction. For such a scenario, it is presumed that the subject confirmation obligations described by the
578 statements within the assertion would be of type, `urn:oasis:names:tc:SAML:1.0:cm:bearer` [\[SAMLBind11\]](#)
579 and that the relying party would validate the assertion in accordance with the processing rules of [\[SAMLCore11\]](#).
580 Particular attention must be paid to the proper validation of the `<saml:AudienceRestrictionCondition>` el-
581 ement which specifies the intended consumer(s) of the assertion. In this case the assertion construction guidance in
582 [Section 6.3.2](#) would apply.

583 An example of a SAML bearer token can be found in [Section 9.4](#).

584 This specification does not limit the types of bearer tokens which can be conveyed to the token forms profiled by
585 [\[wss-sms\]](#), [\[wss-x509\]](#) or [\[wss-saml\]](#). That is, custom tokens or tokens which are subsequently profiled after this
586 specification is finalized could still leverage this mechanism providing the `wsse:ValueType` is understood by the
587 producer and consumer of the token. See the second example in [Section 9.4](#).

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

589 These mechanisms do not protect the integrity, authenticity or confidentiality of the bearer token and thus caution
590 must be taken to not expose the token to unauthorized entities. To secure a message from such threats, one of the
591 mechanisms which support peer entity authentication with integrity and confidentiality protections (see [Section 6.2](#))
592 should be used in conjunction with or instead of an unprotected bearer mechanism.

593 6.3.3.1. Sender Processing Rules

- 594 • The construction and decoration of the `<wsse:Security>` header element MUST adhere to the rules specified in
595 [\[wss-sms\]](#).
- 596 • When the token type is well known or takes an obvious form (e.g. `<wsse:BinarySecurityToken>`), the sender
597 MAY simply insert the token within the `<wsse:Security>` header.
- 598 • When the token type is not well known or in an obvious form, the sender SHOULD wrap the bearer token within
599 a `<wsse:Embedded>` element and make it a child of a `<wsse:SecurityTokenReference>` as described in
600 section 7.4 of [\[wss-sms\]](#).
601 The sender SHOULD indicate the type of the token by specifying the `wsse:ValueType` attribute of the
602 `<wsse:Embedded>` element.
- 603 • This bearer token MUST NOT be referenced by a WS-Security signature in the message.

604 6.3.3.2. Recipient Processing Rules

- 605 • The recipient MUST locate the `<wsse:Security>` element for which it is the target. This MUST adhere to the
606 syntax and processing rules specified in [\[wss-sms\]](#)
- 607 • The recipient MUST locate the bearer token by processing `<wsse:Embedded>` elements within the
608 `<wsse:SecurityTokenReference>` element.
609 The recipient MUST process the token in accordance with the processing rules of the token type as indicated by
610 the `wsse:ValueType` attribute of the `<wsse:Embedded>` element.
- 611 • Alternatively the recipient MAY be able to locate the token by its well known schema type (e.g.
612 `<wsse:BinarySecurityToken>`.)

613 7. Supporting Schema

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

617 7.1. ProxySubject Schema

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

```
621     <xs:element name="ProxySubject" substitutionGroup="saml:Subject"
622               type="saml:SubjectType" />
623
624
```

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

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

630 7.2. ProxyTransitedStatement Schema

631 The <ProxyTransitedStatement> is used to identify an entity which actively participated in the message ex-
632 changes leading up to a given resource access. Its intended usage is twofold. First, it MAY be used by the consumer
633 of authorization data for conveyance to the assertion issuing authority. Second, the assertion issuing authority MAY
634 propagate this information as advice within assertions it subsequently generates.

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

```
636     <xs:annotation>
637       <xs:documentation>ProxyTransitedStatement is a
638         SubjectStatement which MAY carry specific subject confirmation
639         data </xs:documentation>
640     </xs:annotation>
641     <xs:element name="ProxyTransitedStatement"
642               type="saml:SubjectStatementAbstractType" />
643
644
645
```

646 7.3. ProxyInfoConfirmationData Schema

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

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

```
650     <xs:annotation>
651       <xs:documentation>
652         ProxyInfoConfirmationData may be relied upon to
653         corroborate the path information carried in a
654         ProxyTransitedStatement
655     </xs:documentation>
```

```
656 </xs:annotation>
657 <xs:element name="ProxyInfoConfirmationData"
658     type="sec:ProxyInfoConfirmationType" />
659 <xs:complexType name="ProxyInfoConfirmationType">
660     <xs:sequence>
661         <xs:element ref="saml:AssertionIDReference" />
662         <xs:element name="Issuer" type="xs:string" />
663         <xs:element name="IssueInstant" type="xs:dateTime" />
664         <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1" />
665     </xs:sequence>
666     <xs:attribute name="id" type="xs:ID" />
667 </xs:complexType>
668
669
670
```

671 The semantics around the elements are as follows:

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

677 7.4. SessionContext Schema

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

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

```
681 <xs:element name="SessionContext" type="sec:SessionContextType" />
682 <xs:complexType name="SessionContextType">
683     <xs:sequence>
684         <xs:element name="SessionSubject" type="lib:SubjectType" />
685         <xs:element name="ProviderID" type="md:entityIDType" />
686         <xs:element ref="lib:RequestAuthnContext" minOccurs="0" maxOccurs="1" />
687         <!-- The system entity for which this context applies
688             is privacy protect by the SessionSubject -->
689     </xs:sequence>
690     <xs:attribute name="SessionIndex" type="xs:string" use="optional" />
691     <xs:attribute name="AuthenticationInstant" type="xs:dateTime" use="required" />
692     <xs:attribute name="AssertionIssueInstant" type="xs:dateTime" use="required" />
693 </xs:complexType>
694
695
696
```

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

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

- 706 • The <ProviderID> element is a URI which indicates the entity (session holder) with which the session applies.
707 <ProviderID> MUST correspond with the constraints for the <entityIDType> data type as specified in
708 [\[LibertyMetadata\]](#).
- 709 • The <ac:AuthnContext> element as specified in [\[LibertyAuthnContext\]](#) depicts the authentication context
710 which the session holder relied upon to create the session.
- 711 • The SessionIndex attribute is used as an aid when in managing multiple sessions with a Principal. This
712 attribute represents the same thing as the attribute of the same name in the lib:AuthenticationStatement
713 [\[LibertyProtSchema\]](#)
- 714 • The AuthenticationInstant attribute describes the time at which the subject of the session authenticated to
715 an authority.
- 716 • The AssertionIssueInstant attribute is the time at which the assertion issuer generated an assertion for the
717 provider.

718 7.5. SessionContextStatement Schema

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

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

```
722 <xs:element name="SessionContextStatement"  
723           type="sec:SessionContextStatementType"  
724           substitutionGroup="saml:SubjectStatement"/>  
725  
726 <xs:complexType name="SessionContextStatementType">  
727   <xs:complexContent>  
728     <xs:extension base="saml:SubjectStatementAbstractType">  
729       <xs:sequence>  
730         <!-- This is the name of the proxy and it SHOULD carry  
731            SubjectConfirmation information to authorize the  
732            ProxySubject to act on behalf of the  
733            Subject inherited from  
734            SubjectStatementAbstractType -->  
735         <xs:element name="ProxySubject"  
736                   type="saml:SubjectType" minOccurs="0"/>  
737         <xs:element ref="sec:SessionContext"/>  
738       </xs:sequence>  
739     </xs:extension>  
740   </xs:complexContent>  
741 </xs:complexType>  
742  
743  
744
```

745 The <SessionContextStatement> derives from <saml:SubjectStatementAbstractType> and MUST ad-
746 here to the following constraints:

- 747 • It MAY include a <ProxySubject> element which adheres to the structure and semantics described in this
748 specification
- 749 • It MUST include a <SessionContext> element which adheres to the structure and semantics described in this
750 specification

751 7.6. ResourceAccessStatement Schema

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

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

```
755 <xs:element name="ResourceAccessStatement"  
756 type="sec:ResourceAccessStatementType"  
757 substitutionGroup="saml:SubjectStatement" />  
758  
759 <xs:complexType name="ResourceAccessStatementType">  
760 <xs:complexContent>  
761 <xs:extension base="saml:SubjectStatementAbstractType">  
762 <xs:sequence>  
763 <xs:group ref="disco:ResourceIDGroup" />  
764 <xs:sequence minOccurs="0">  
765 <!-- This is the name of the proxy and it SHOULD carry  
766 SubjectConfirmation information to authorize the  
767 ProxySubject to act on behalf of the  
768 Subject inherited from  
769 SubjectStatementAbstractType -->  
770 <xs:element name="ProxySubject" type="saml:SubjectType" />  
771 <xs:element ref="sec:SessionContext" minOccurs="0" />  
772 </xs:sequence>  
773 </xs:sequence>  
774 </xs:extension>  
775 </xs:complexContent>  
776 </xs:complexType>  
777  
778
```

779 The <ResourceAccessStatement> derives from <saml:SubjectStatementAbstractType> and MUST ad-
780 here to the following constraints:

- 781 • It MUST contain either a <disco:ResourceID> or a <disco:EncryptedResourceID> element which ad-
782 heres to the processing rules of [\[LibertyDisco\]](#)
- 783 • It MAY include a <ProxySubject> element which adheres to the structure and semantics described in this
784 specification
- 785 • It MAY include a <SessionContext> element which adheres to the structure and semantics described in this
786 specification

787 **8. Message Authorization Model**

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

794 **8.1. Authorization Mechanism Overview (Informative)**

795 The authorization mechanism defined by this specification formalizes the generation and conveyance of authorization
796 information. In support of this mechanism a Trusted Third Party (TTP) may be relied upon to act as a Policy Decision
797 Point (PDP) and potentially a coarse grained Policy Enforcement Point (PEP) to facilitate the exchange of resource
798 access information to the relying party. As a PDP, the Trusted Third Party would adhere to the coarse access policies
799 of the relying party insofar as ensuring which entities may attempt to access a given resource. This requires strong
800 assurance as to the authenticity of a peer subject. Given the reliance of authorization upon authentication, this model
801 aids in disseminating subject confirmation obligations, identity information and access authorization data.

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

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

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

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

817 **8.2. Authorization Mechanism**

818 It is RECOMMENDED that this mechanism utilize the Web Services Security SAML Profile [\[wss-saml\]](#) as the
819 means by which the message sender authenticates to the recipient. Each communicating peer performs message
820 level authentication by demonstrating proof of possession of a subject confirmation key. The assertion issuer binds
821 the subject confirmation key to the assertion by signing the assertion. This attestation assures the consumer of the
822 assertion that the subject confirmation key is that of the intended sender. Thus the senders subject confirmation key
823 can be recognized by the recipient as belonging to the remote peer. Subsequent to the authentication of the sender the
824 recipient can leverage this knowledge in support of the authorization model described below.

825 The authorization model supports the issuance of assertions that convey information regarding the resource to be
826 accessed, the entity attempting to access the resource, the mechanism by which the accessing entity must use to
827 demonstrate its identity to the recipient and the ability for the accessing entity to access the resource on behalf of
828 another system entity. This latter facility suggests the need to verify two distinct identities in a given resource access
829 message, the sender identity and the invocation identity. Thus the authorization model supports a constrained proxy
830 mechanism that permits a proxy (the sender) to access the resource on behalf of some other system entity.

831 **8.3. Authorization Data Generation**

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

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

840 **8.3.1. Processing Rules**

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

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

- 847 • The assertion **MUST** include at most one of the following statements; `<SessionContextStatement>` or a
848 `<ResourceAccessStatement>`.
- 849 • The `<saml:Subject>` element **MUST** describe the invocation identity. The invocation identity **MUST** be either
850 that of the sender or another system entity on whose behalf the sender (as a proxy) is authorized to act.
- 851 • The assertion **MUST** describe the invocation identity within the `<saml:Subject>` element of the base statement.
- 852 • When the invocation identity is that of the sender the `<saml:Subject>` element of the base statement type **MUST**
853 be constructed as follows:
854 The `<saml:Subject>` element **MUST** include a `<saml:SubjectConfirmation>` element with a
855 `<saml:ConfirmationMethod>` of `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`.
856 The subject confirmation key **MUST** be specified within the `<saml:Subject>` element by including a
857 `<ds:KeyInfo>` element in the `<saml:SubjectConfirmation>` element.
- 858 • When the invocation identity is **NOT** that of the sender the `<saml:Subject>` element of the base statement type
859 **MUST** be constructed as follows:
860 The assertion **SHOULD** protect the privacy of the named entity with a `<lib:EncryptedNameIdentifier>`
861 element.
862 The authority **MUST** include a `<saml:SubjectConfirmation>` element in this subject and spec-
863 ify that the sender will vouch for the subject by indicating a `<saml:ConfirmationMethod>` of
864 `urn:oasis:names:tc:SAML:1.0:cm:sender-vouches`.
865 Additionally, the assertion **MUST** identify the proxy and its confirmation requirements by including a
866 `<ProxySubject>` element.. This element **MUST** be constructed in accordance with the constraints defined in
867 [ProxySubject Schema \(Section 7.1\)](#). **OPTIONALLY**, the assertion issuer **MAY** include information that assists in
868 building a chain of transited proxies. It is **RECOMMENDED** that the `<saml:Advice>` element be decorated
869 with a `<saml:AssertionIDReference>` which is a reference to the assertion bearing it. Also as the chain
870 builds the assertion should be augmented with `<ProxyTransitedStatement>`. The issuer should include a
871 `<ProxyTransitedStatement>` for each proxy that has participated in the progression of assertion issuance.
872 See [Proxy Chaining \(Section 8.3.2\)](#) for a recommendation on constructing the proxy chain.

- 873 • For statements of type `<ResourceAccessStatement>` the assertion issuing authority MUST adhere to the
874 construction constraints described in [ResourceAccessStatement Schema \(Section 7.6\)](#).
875 The assertion issuing authority MAY describe the authentication status of the interacting party by including a
876 `<SessionContext>` element. The `<SessionContext>` element MUST adhere to the construction constraints
877 described in [SessionContext Schema \(Section 7.6\)](#).
- 878 • For statements of type `<SessionContextStatement>` the assertion issuing authority MUST adhere to the
879 construction constraints described in [SessionContextStatement Schema \(Section 7.5\)](#) and [SessionContext Schema](#)
880 [\(Section 7.6\)](#).
- 881 • The assertion MUST be signed by the assertion issuing authority in accordance with the signing requirements
882 specified in [\[SAMLCore11\]](#).

883 8.3.2. Proxy Chaining

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

886 It is presumed that when a system entity interacts with the assertion issuing authority, the system entity will include
887 a claim that contains a `<ProxyTransitedStatement>` bearing `<ProxyInfoConfirmationData>`. This claim
888 SHOULD be in the form of a SAML assertion carried as a security token within the security header of the request to
889 the assertion issuing authority.

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

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

```
899 <saml:Assertion
900   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
901   MajorVersion="1" MinorVersion="0"
902   AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
903   Issuer="authority.example.com"
904   IssueInstant="2004-04-01T16:58:33.173Z">
905
906   <saml:Advice>
907     <saml:AssertionIDReference>refers to this assertion</saml:AssertionIDReference>
908     <saml:Assertion>
909       <!-- This statement reflects the first proxy transited -->
910       <ProxyTransitedStatement>
911         <Issuer>authority.example.com</Issuer>
912         <IssueInstant>2004-04-01T16:58:30.173Z</IssueInstant>
913         <saml:AssertionIDReference>
914           <!-- refers to assertion first.example.com confirmed to
915             authority.example.com to get an assertion which named
916             first.example.com the ProxySubject. -->
917         </saml:AssertionIDReference>
918         <saml:Subject>
919           <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
920             http://first.example.com/
921           </saml:NameIdentifier>
922         </saml:Subject>
923       </ProxyTransitedStatement>
```

```

924     <ProxyTransitedStatement>
925         <Issuer>authority.example.com</Issuer>
926         <IssueInstant>2004-04-01T16:58:32.173Z</IssueInstant>
927         <saml:AssertionIDReference>
928             <!-- refers to assertion second.example.com confirmed to
929                 authority.example.com to get an assertion which named
930                 second.example.com the ProxySubject. -->
931         </saml:AssertionIDReference>
932         <saml:Subject>
933             <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
934                 http://second.example.com/
935             </saml:NameIdentifier>
936         </saml:Subject>
937     </ProxyTransitedStatement>
938 </saml:Assertion>
939 </saml:Advice>
940
941 <!-- By placing an audience restriction on the assertion we
942      can limit the scope of which entity should consume
943      the information in the assertion. -->
944
945 <saml:Conditions
946     NotBefore="2004-04-01T16:57:20Z"
947     NotOnOrAfter="2004-04-01T21:42:43Z">
948     <saml:AudienceRestrictionCondition>
949         <saml:Audience>http://fourth.example.com</saml:Audience>
950     </saml:AudienceRestrictionCondition>
951 </saml:Conditions>
952
953 <!-- The AuthenticationStatement carries information
954      that describes the identity of the entity this assertion
955      was issued to (the Subject) and the method the Subject
956      authenticated to the assertion issuing authority -->
957 <saml:AuthenticationStatement
958     AuthenticationMethod="urn:ietf:rfc:2246"
959     AuthenticationInstant="2004-04-01T16:57:30.000Z">
960     <saml:Subject>
961         <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
962             http://third.example.com/
963         </saml:NameIdentifier>
964     </saml:Subject>
965 </saml:AuthenticationStatement>
966
967 <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
968     <saml:Subject>
969         <!-- the name identifier of the interacting entity -->
970         <saml:NameIdentifier ID="#abesyd"
971             Format="urn:liberty:iff:nameid:encrypted">
972             HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhWbDFNDELgscSXZ5Ekw
973             A23B45...C569UXR3==
974         </saml:NameIdentifier>
975         <SubjectConfirmation>
976             <!-- The confirmed sender of this message vouches for the
977                 identity carried in this subject. -->
978             <ConfirmationMethod>
979                 urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
980             </ConfirmationMethod>
981         </SubjectConfirmation>
982     </saml:Subject>
983     <ProxySubject>
984         <!-- the name identifier of the sender -->
985         <NameIdentifier format="urn:liberty:iff:nameid:entityID">
986             http://third.example.com/
987         </NameIdentifier>
988         <SubjectConfirmation>
989             <ConfirmationMethod>
990                 urn:oasis:names:tc:SAML:1.0:cm:holder-of-key

```

```
991         </ConfirmationMethod>
992         <!-- This keyinfo is the key by which the sender must prove
993             possession. -->
994         <ds:KeyInfo>
995             <ds:KeyName>
996                 CN=third.example.com,OU=Services R US,O=Service Nation,...
997             </ds:KeyName>
998             <ds:KeyValue>...</ds:KeyValue>
999         </ds:KeyInfo>
1000     </SubjectConfirmation>
1001 </ProxySubject>
1002 </ResourceAccessStatement>
1003 <!-- signature by the authority over the assertion -->
1004 <ds:Signature>...</ds:Signature>
1005 </saml:Assertion>
1006
1007
```

1008 **8.4. Presenting Authorization Data**

1009 Interactions with identity-based web services may rely on the conveyance of authorization information. In general, a
1010 trusted authority issues the authorization data. In such a setting the authorization information would be sent along with
1011 the identity-based web service request to the recipient. See [Authorization Data Generation \(Section 8.3\)](#) for details as
1012 to how this data is acquired and formulated.

1013 **8.4.1. Processing Rules**

- 1014 • The sender **MUST** authenticate to the recipient using one of the authentication mechanisms described in [Message](#)
1015 [Authentication \(Section 6.3\)](#).
1016 It is **RECOMMENDED** that the sender authenticate using the [SAML Assertion Message Authentication](#) and
1017 specifically conform to the processing rules specified in ([Section 6.3.2.1](#)).
- 1018 • In accordance with the above requirement the sender **MUST** include the resultant XML signature in a
1019 `<ds:Signature>` element. The signature element **SHOULD** be decorated with a `ds:KeyInfo` which **MUST**
1020 resolve to the subject confirmation key.
1021 To assist the receiver in processing the authorization information, the sender decorates the `<ds:Signature>` with
1022 a `<ds:KeyInfo>` which **SHOULD** include a `<wsse:SecurityTokenReference>` element in accordance with
1023 the [\[wss-sms\]](#).

1024 **8.5. Consuming Authorization Data**

1025 A recipient which exposes a resource typically makes access control decisions based on the invocation identity.
1026 Additionally the recipient may also predicate access control policies upon the sender identity. The semantics of
1027 resource access authorization are described in [Presenting Authorization Data \(Section 8.4\)](#).

1028 The recipient determines the invocation identity by inspecting the `<saml:Subject>` element and the proxy identity
1029 by inspecting the `<ProxySubject>` if present. When the `<ProxySubject>` element is present it also describes the
1030 subject confirmation obligation of the proxy. Providing both the invocation identity and the proxy identity enables
1031 the recipient to tailor authorization policy to a finer degree of granularity. That is, the recipient generally uses the
1032 invocation identity to make its authorization decisions and potentially determine whether the proxy is permitted to
1033 access the resource on behalf of said invocation identity.

1034 **8.5.1. Processing Rules**

- 1035 • The recipient **MUST** authenticate the sender using one of the mechanisms described in [Authentication Mecha-](#)
1036 [nisms](#).
1037 It is **RECOMMENDED** that the sender authenticate using the [SAML Assertion Message Authentication](#) and
1038 specifically conform to the processing rules specified in ([Section 6.3.2.2](#)).
- 1039 • The recipient **MUST** locate the `<saml:Assertion>` (security token) which conferred the subject confirmation
1040 key relied upon for sender authentication.
1041 The recipient **MUST** corroborate that the bound subject confirmation key is the same key used to authenticate the
1042 communicating peer.
- 1043 • The recipient **MUST** determine that it trusts the authority which signed the `<saml:Assertion>`.
1044 The recipient **MUST** validate the signature of the `<saml:Assertion>`. The recipient **SHOULD** validate the trust
1045 semantics of the signing key, as appropriate to the risk of incorrect authentication.

1046 9. Identity-Based Web Service Examples (Informative)

1047 The following are examples demonstrating the various use cases supported by this specification.

1048 9.1. Conveyance of Sender as Invocation Identity

1049 Recall that the authorization model describes a mechanism for an assertion issuing authority to bind the invocation
1050 identity to an assertion by specifying the invocation identity within the <saml:Subject>. The following example
1051 demonstrates the construction of a <saml:Subject> to accomplish this binding:

```
1052 <Subject>
1053 <!-- the name identifier of the sender -->
1054 <NameIdentifier format="urn:liberty:iff:nameid:entityID">
1055   http://serviceprovider.com/
1056 </NameIdentifier>
1057 <SubjectConfirmation>
1058   <ConfirmationMethod>
1059     urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
1060   </ConfirmationMethod>
1061   <!-- This keyinfo is the key by which the sender must prove
1062     possession. Note the KeyName MAY but is NOT REQUIRED to
1063     match the above NameIdentifier -->
1064   <ds:KeyInfo>
1065     <ds:KeyName>
1066       CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
1067     </ds:KeyName>
1068     <ds:KeyValue>...</ds:KeyValue>
1069   </ds:KeyInfo>
1070 </SubjectConfirmation>
1071 </Subject>
1072
```

1073 Contents in the above example worth particular mention include:

- 1074 • The assertion issuing authority specifies the <saml:ConfirmationMethod> with the value of
1075 urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
- 1076 • The assertion issuing authority included a <ds:KeyInfo> element to indicate which key the <saml:Subject>
1077 must demonstrate possession of to meet the above confirmation obligation.

1078 What follows is a more complete example that depicts a request to access an identity-based web service that carries
1079 authorization data to the recipient such that the sender identity and the invocation identity are the same. The resource
1080 for which the sender is attempting to access is described in a ResourceAccessStatement bound to the assertion.

1081 The interpretation of the assertion can be expressed as follows:

1082 According to the policies of the assertion issuing authority the sender must adhere to the confirmation requirements
1083 expressed in the encapsulated ResourceAccessStatement. To meet the obligation the message sender;

- 1084 • MUST successfully demonstrate possession of the key referenced in the ResourceAccessStatement's SubjectCon-
1085 firmation element which is bound to the Assertion
- 1086 • AND the message receiver has a trusted path from the signing certificate to an issuing Certification Authority.

1087 Note that, while the assertion associates a subject's name with a key, this association is made as a means to indicate
1088 the authorization of that subject, acting with that key, to invoke a service. This facility, incorporated for authorization
1089 purposes, serves a distinct and complementary function to the binding between subject and key, which the subject's
1090 certificate accomplishes for authentication purposes.

```
1091 <?xml version="1.0" encoding="UTF-8"?>
1092 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1093     xmlns:sb="urn:liberty:sb:2003-08"
1094     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1095     xmlns:sec="urn:liberty:sec:2003-08">
1096
1097   <s:Header>
1098     <sb:Correlation s:mustUnderstand="1"
1099         id="A13454...245"
1100         actor="http://schemas.../next"
1101         messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
1102         timestamp="2112-03-15T11:12:12Z"/>
1103
1104   <wsse:Security>
1105     <saml:Assertion
1106         xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1107         MajorVersion="1" MinorVersion="0"
1108         AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1109         Issuer="idp.example.com"
1110         IssueInstant="2004-04-01T16:58:33.173Z">
1111
1112       <!-- By placing an audience restriction on the assertion we
1113           can limit the scope of which entity should consume
1114           the information in the assertion. -->
1115
1116       <saml:Conditions
1117           NotBefore="2004-04-01T16:57:20Z"
1118           NotOnOrAfter="2004-04-01T21:42:43Z">
1119         <saml:AudienceRestrictionCondition>
1120           <saml:Audience>http://webserviceprovider.com</saml:Audience>
1121         </saml:AudienceRestrictionCondition>
1122       </saml:Conditions>
1123
1124       <!-- The AuthenticationStatement carries information
1125           that describes the identity of the entity this assertion
1126           was issued to (the Subject) and the method the Subject
1127           authenticated to the assertion issuing authority -->
1128       <saml:AuthenticationStatement
1129           AuthenticationMethod="urn:ietf:rfc:2246"
1130           AuthenticationInstant="2004-04-01T16:57:30.000Z">
1131         <saml:Subject>
1132           <saml:NameIdentifier
1133               format="urn:liberty:iff:nameid:entityID"
1134               NameQualifier="http://AffiliationStation.com/">
1135             http://serviceprovider.com/
1136           </saml:NameIdentifier>
1137         </saml:Subject>
1138       </saml:AuthenticationStatement>
1139
1140       <!-- The Subject of the ResourceAccessStatement specifies use of
1141           holder-of-key which indicates the subject of the statement
1142           is the sender of the message and is to be treated as the
1143           invocation identity for any access control decisions -->
1144       <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
1145         <saml:Subject>
1146           <!-- the name identifier of the sender -->
1147           <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
1148             http://serviceprovider.com/
1149           </saml:NameIdentifier>
1150         <saml:SubjectConfirmation>
1151           <saml:ConfirmationMethod>
1152             urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
```

```
1153     </saml:ConfirmationMethod>
1154     <!-- This keyinfo is the key by which the sender must prove
1155           possession. Note the KeyName MAY but is NOT REQUIRED to
1156           match the above NameIdentifier -->
1157     <ds:KeyInfo>
1158       <ds:KeyName>
1159         CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
1160       </ds:KeyName>
1161       <ds:KeyValue>...</ds:KeyValue>
1162     </ds:KeyInfo>
1163   </saml:SubjectConfirmation>
1164 </saml:Subject>
1165
1166   <!-- This ResourceID describes the resource for which
1167         the sender is attempting to access. -->
1168   <disco:ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>
1169 </ResourceAccessStatement>
1170 <!-- signature by the authority over the assertion -->
1171 <ds:Signature>...</ds:Signature>
1172 </saml:Assertion>
1173 <!-- this is the signature the sender generated to demonstrate holder-of-key
1174       the signature should cover the isf header and body-->
1175 <ds:Signature>
1176   <ds:SignedInfo>
1177     ...
1178     <ds:Reference URI="#A13454...245">
1179       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1180       <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1181     </ds:Reference>
1182     <ds:Reference URI="#MsgBody">
1183       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1184       <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1185     </ds:Reference>
1186   </ds:SignedInfo>
1187 </ds:KeyInfo>
1188   <wsse:SecurityTokenReference>
1189     <wsse:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1190       ValueType="saml:Assertion" />
1191   </wsse:SecurityTokenReference>
1192 </ds:KeyInfo>
1193 <ds:SignatureValue>
1194   HJJWbvqW9E84vJVQkjJLLA6nNvBX7mY00TzhwBdFNDElgsCSXZ5Ekw==
1195 </ds:SignatureValue>
1196 </ds:Signature>
1197 </wsse:Security>
1198 </s:Header>
1199 <s:Body wsu:Id="MsgBody">
1200   <pp:Modify>
1201     <!-- this is an ID-SIS-PP Modify message -->
1202   </pp:Modify>
1203 </s:Body>
1204 </s:Envelope>
1205
1206
```

1207 9.2. Conveyance of Sender as Proxy

1208 This authorization model supports environments where the sender of a message is acting as a proxy on behalf of
1209 another system entity. The following example demonstrates the construction of a <ProxySubject> to accomplish
1210 this binding:

```
1211 <ProxySubject>
1212   <!-- the name identifier of the proxy -->
1213   <NameIdentifier format="urn:liberty:iff:nameid:entityID">
1214     http://serviceprovider.com/
1215   </NameIdentifier>
```

```
1216 <SubjectConfirmation>
1217   <ConfirmationMethod>
1218     urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
1219   </ConfirmationMethod>
1220   <!-- This keyinfo is the key by which the proxy must prove
1221     possession. Note the KeyName MAY but is NOT REQUIRED to
1222     match the above NameIdentifier -->
1223   <ds:KeyInfo>
1224     <ds:KeyName>
1225       CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
1226     </ds:KeyName>
1227     <ds:KeyValue>...</ds:KeyValue>
1228   </ds:KeyInfo>
1229 </SubjectConfirmation>
1230 </ProxySubject>
1231
```

1232 Contents in the above example worth particular mention include:

- 1233 • The assertion issuing authority generated assertion indicates that the <saml:Subject> element is that of the
1234 system entity on whose behalf the sender is proxying.
- 1235 • The assertion includes a <ProxySubject> element so as to convey the identity of the proxy to the relying party.
- 1236 • The <ProxySubject> element specifies the <saml:ConfirmationMethod> with a value of
1237 urn:oasis:names:tc:SAML:1.0:cm:holder-of-key.
- 1238 • The <saml:SubjectConfirmation> element includes a <ds:KeyInfo> element which indicates which key
1239 the proxy must prove possession of to meet the confirmation obligation.

1240 The following example, is a more complete demonstration of a request invoking an identity-based web service which
1241 carries authorization data to the recipient such that the sender can act as a proxy on behalf of the entity described in
1242 the subject of the resource access statement.

```
1243 <?xml version="1.0" encoding="UTF-8"?>
1244 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1245   xmlns:sb="urn:liberty:sb:2003-08"
1246   xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1247   xmlns:sec="urn:liberty:sec:2003-08">
1248
1249   <s:Header>
1250     <sb:Correlation s:mustUnderstand="1"
1251       id="A13454...245"
1252       actor="http://schemas.../next"
1253       messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
1254       timestamp="2112-03-15T11:12:12Z"/>
1255   <wsse:Security>
1256     <saml:Assertion
1257       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1258       MajorVersion="1" MinorVersion="0"
1259       AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1260       Issuer="idp.example.com"
1261       IssueInstant="2004-04-01T16:58:33.173Z">
1262
1263       <!-- By placing an audience restriction on the assertion we
1264         can limit the scope of which entity should consume
1265         the information in the assertion. -->
1266
1267       <saml:Conditions
1268         NotBefore="2004-04-01T16:57:20Z"
1269         NotOnOrAfter="2004-04-01T21:42:43Z">
1270         <saml:AudienceRestrictionCondition>
1271           <saml:Audience>http://webserviceprovider.com</saml:Audience>

```

```

1272     </saml:AudienceRestrictionCondition>
1273 </saml:Conditions>
1274
1275 <!-- The AuthenticationStatement carries information
1276      that describes the identity of the entity this assertion
1277      was issued to (the Subject) and the method the Subject
1278      authenticated to the assertion issuing authority -->
1279 <saml:AuthenticationStatement
1280   AuthenticationMethod="urn:ietf:rfc:2246"
1281   AuthenticationInstant="2004-04-01T16:57:30.000Z">
1282   <saml:Subject>
1283     <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
1284       http://serviceprovider.com/
1285     </saml:NameIdentifier>
1286   </saml:Subject>
1287 </saml:AuthenticationStatement>
1288
1289 <!-- Subject of the ResourceAccessStatement specifies a
1290      confirmation method of sender-vouches. This indicates that the
1291      subject of the statement is being vouched for by the sender of
1292      the message. The sender identity is conveyed in the
1293      ProxySubject element. In order for the ProxySubject to vouch
1294      for the Subject the ProxySubject must meet the holder-of-key
1295      confirmation obligation. The proxy does so by demonstrating
1296      possession of the described key. In this scenario the Subject of
1297      the message is to be treated as the invocation identity (for
1298      any access control decisions) -->
1299
1300 <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
1301 <saml:Subject>
1302   <!-- the name identifier of the interacting entity -->
1303   <saml:NameIdentifier ID="#abesyd"
1304     Format="urn:liberty:iff:nameid:encrypted">
1305     HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TzhwBdFNDElgsCSXZ5 Ekw
1306     A23B45...C569UXR3==
1307   </saml:NameIdentifier>
1308   <SubjectConfirmation>
1309     <!-- The confirmed sender of this message vouches for the
1310          identity carried in this subject. -->
1311     <ConfirmationMethod>
1312       urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
1313     </ConfirmationMethod>
1314   </SubjectConfirmation>
1315 </saml:Subject>
1316 <ProxySubject>
1317   <!-- the name identifier of the sender -->
1318   <NameIdentifier format="urn:liberty:iff:nameid:entityID">
1319     http://serviceprovider.com/
1320   </NameIdentifier>
1321   <SubjectConfirmation>
1322     <ConfirmationMethod>
1323       urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
1324     </ConfirmationMethod>
1325     <!-- This keyinfo is the key by which the sender must prove
1326          possession. -->
1327     <ds:KeyInfo>
1328       <ds:KeyName>
1329         CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
1330       </ds:KeyName>
1331       <ds:KeyValue>...</ds:KeyValue>
1332     </ds:KeyInfo>
1333   </SubjectConfirmation>
1334 </ProxySubject>
1335
1336 <!-- This ResourceID describes the resource for which
1337      the sender is attempting to access. -->
1338 <disco:ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>

```

```

1339     </ResourceAccessStatement>
1340     <!-- signature by the authority over the assertion -->
1341     <ds:Signature>...</ds:Signature>
1342 </saml:Assertion>
1343 <!-- this is the signature the sender generated to demonstrate holder-of-key
1344     the signature should cover the isf header and body-->
1345 <ds:Signature>
1346 <ds:SignedInfo>
1347     ...
1348 <ds:Reference URI="#A13454...245">
1349 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1350 <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1351 </ds:Reference>
1352 <ds:Reference URI="#MsgBody">
1353 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1354 <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1355 </ds:Reference>
1356 </ds:SignedInfo>
1357 <ds:KeyInfo>
1358 <wsse:SecurityTokenReference>
1359 <wsse:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1360 ValueTypes="saml:Assertion" />
1361 </wsse:SecurityTokenReference>
1362 </ds:KeyInfo>
1363 <ds:SignatureValue>
1364 HJJWbvqW9E84vJVQkjJLLA6nNvBX7mY00TZhwBdFNDElgs cSXZ5Ekw==
1365 </ds:SignatureValue>
1366 </ds:Signature>
1367 </wsse:Security>
1368 </s:Header>
1369 <s:Body wsu:Id="MsgBody">
1370 <pp:Modify>
1371 <!-- this is an ID-SIS-PP Modify message -->
1372 </pp:Modify>
1373 </s:Body>
1374 </s:Envelope>
1375

```

1376 9.3. Session Context of the Interacting Entity

1377 Access to resources exposed by a service instance are nominally restricted by access control policy enforced by the
1378 entity hosting the resource. Additionally, the policy information, enforcement and decision points may be distributed
1379 across multiple system entities. Authorization to access a resource may require that the entity interacting (e.g. browser
1380 principal) with another entity (e.g. service consumer) have an active authenticated session.

1381 To facilitate this scenario the trusted authority may supply authorization data that conveys the session status of the
1382 interacting entity.

1383 The following example is similar to the above example with the addition of conveying session context of the entity
1384 (principal) interacting with the message sender.

```

1385 <?xml version="1.0" encoding="UTF-8"?>
1386 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1387     xmlns:sb="urn:liberty:sb:2003-08"
1388     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1389     xmlns:sec="urn:liberty:sec:2003-08">
1390
1391 <s:Header>
1392 <sb:Correlation s:mustUnderstand="1"
1393     id="A13454...245"
1394     actor="http://schemas.../next"
1395     messageID="uuid:efefefef-aaaa-ff ff-cccc-eeeeffffbbbb"
1396     timestamp="2112-03-15T11:12:12Z"/>
1397 <wsse:Security>

```

```

1398 <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1399   AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1400   Issuer="idp.example.com"
1401   IssueInstant="2004-04-01T16:58:33.173Z">
1402
1403   <!-- By placing an audience restriction on the assertion we
1404     can limit the scope of which entity should consume
1405     the information in the assertion. -->
1406
1407   <saml:Conditions
1408     NotBefore="2004-04-01T16:57:20Z"
1409     NotOnOrAfter="2004-04-01T21:42:43Z">
1410     <saml:AudienceRestrictionCondition>
1411       <saml:Audience>http://webserviceprovider.com</saml:Audience>
1412     </saml:AudienceRestrictionCondition>
1413   </saml:Conditions>
1414
1415   <!-- The AuthenticationStatement carries information
1416     that describes the identity of the entity this assertion
1417     was issued to (the Subject) and the method the Subject
1418     authenticated to the assertion issuing authority -->
1419   <saml:AuthenticationStatement
1420     AuthenticationMethod="urn:ietf:rfc:2246"
1421     AuthenticationInstant="2004-04-01T16:57:30.000Z">
1422     <saml:Subject>
1423       <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
1424         http://serviceprovider.com/
1425       </saml:NameIdentifier>
1426     </saml:Subject>
1427   </saml:AuthenticationStatement >
1428
1429   <!-- subject of the ResourceAccessStatement specifies use of
1430     holder-of-key which indicates the subject of the statement
1431     is the sender of the message and is to be treated as the
1432     invocation identity for any access control decisions -->
1433   <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
1434     <Subject>
1435       <!-- the name identifier of the sender -->
1436       <NameIdentifier format="urn:liberty:iff:nameid:entityID">
1437         http://serviceprovider.com/
1438       </NameIdentifier>
1439       <SubjectConfirmation>
1440         <ConfirmationMethod>
1441           urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
1442         </ConfirmationMethod>
1443         <!-- This keyinfo is the key by which the sender must prove
1444           possession. Note the KeyName MAY but is NOT REQUIRED to
1445           match the above NameIdentifier -->
1446         <ds:KeyInfo>
1447           <ds:KeyName>
1448             CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
1449           </ds:KeyName>
1450           <ds:KeyValue>...</ds:KeyValue>
1451         </ds:KeyInfo>
1452       </SubjectConfirmation>
1453     </Subject>
1454
1455     <!-- This ResourceID describes the resource for which
1456     the sender is attempting to access. -->
1457     <disco:ResourceID>http://foo.com/d0CQF8e1JTDLmzEo</disco:ResourceID>
1458     <!-- The session context of the entity interacting with the
1459     request sender -->
1460     <SessionContext xmlns="urn:liberty:sec:2003-08"
1461       AuthenticationInstant="" AssertionIssueInstant="">
1462       <SessionSubject>
1463         <saml:NameIdentifier ID="#abesyd"
1464         Format="urn:liberty:iff:nameid:encrypted">

```

```

1465         HJJWbvqW9E84vJVQkjLLA6nNvBX7mY00TZhwBdFNDEl gscSXZ5Ekw
1466         A23B45C569UkjjLLA6nNvBX7mY00TZhwBd FNDEl gscSXZ5Ekw89XR3==
1467         </saml:NameIdentifier>
1468     </SessionSubject>
1469     <AuthnContext
1470         xmlns="urn:liberty:ac:2003-08">
1471         ...
1472     </AuthnContext>
1473     <ProviderID>http://serviceprovider.com/</ProviderID>
1474 </SessionContext>
1475 </ResourceAccessStatement>
1476 <!-- signature by the authority over the assertion -->
1477 <ds:Signature>...</ds:Signature>
1478 </saml:Assertion>
1479 <!-- this is the signature the sender generated to demonstrate holder-of-key
1480     the signature should cover the isf header and body-->
1481 <ds:Signature>
1482 <ds:SignedInfo>
1483     ...
1484     <ds:Reference URI="#A13454...245">
1485     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1486     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1487     </ds:Reference>
1488     <ds:Reference URI="#MsgBody">
1489     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1490     <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1491     </ds:Reference>
1492 </ds:SignedInfo>
1493 <ds:KeyInfo>
1494     <wsse:SecurityTokenReference>
1495     <wsse:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1496 ValueTypes="saml:Assertion" />
1497     </wsse:SecurityTokenReference>
1498 </ds:KeyInfo>
1499 <ds:SignatureValue>
1500     HJJWbvqW9E84vJVQkjLLA6nNvBX7mY00TZhwBdFNDEl gscSXZ5Ekw==
1501 </ds:SignatureValue>
1502 </ds:Signature>
1503 </wsse:Security>
1504 </s:Header>
1505 <s:Body wsu:Id="MsgBody">
1506     <pp:Modify>
1507     <!-- this is an ID-SIS-PP Modify message -->
1508     </pp:Modify>
1509 </s:Body>
1510 </s:Envelope>
1511

```

1512 9.4. SAML Bearer Token Message Authentication

1513 The following example demonstrates this message authentication mechanism by supplying a SAML security token
1514 [\[wss-saml\]](#) in the security header.

```

1515 <?xml version="1.0" encoding="UTF-8"?>
1516 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1517     xmlns:sb="urn:liberty:sb:2003-08"
1518     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1519     xmlns:sec="urn:liberty:sec:2003-08">
1520
1521     <s:Header>
1522     <sb:Correlation s:mustUnderstand="1"
1523         id="A13454...245"
1524         actor="http://schemas.../next"
1525         messageID="uuid:efefefef-aaaa-ffff-cc cc-eeeeffffbbbb"
1526         timestamp="2112-03-15T11:12:12Z"/>
1527

```



```

1528 <wsse:Security>
1529 <saml:Assertion
1530   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1531   MajorVersion="1" MinorVersion="0"
1532   AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1533   Issuer="idp.example.com"
1534   IssueInstant="2004-04-01T16:58:33.173Z">
1535
1536   <!-- By placing an audience restriction on the assertion we
1537         can limit the scope of which entity should consume
1538         the information in the assertion. -->
1539
1540   <saml:Conditions
1541     NotBefore="2004-04-01T16:57:20Z"
1542     NotOnOrAfter="2004-04-01T21:42:43Z">
1543     <saml:AudienceRestrictionCondition>
1544       <saml:Audience>http://webserviceprovider.com</saml:Audience>
1545     </saml:AudienceRestrictionCondition>
1546   </saml:Conditions>
1547
1548   <!-- The AuthenticationStatement carries information
1549         that describes the identity of the entity this assertion
1550         was issued to (the Subject) and the method the Subject
1551         authenticated to the assertion issuing authority -->
1552
1553   <saml:AuthenticationStatement
1554     AuthenticationMethod="urn:ietf:rfc:2246"
1555     AuthenticationInstant="2004-04-01T16:57:30.000Z">
1556     <saml:Subject>
1557       <saml:NameIdentifier
1558         format="urn:liberty:iff:nameid:entityID"
1559         NameQualifier="http://AffiliationStation.com/">
1560         http://serviceprovider.com/
1561       </saml:NameIdentifier>
1562     </saml:Subject>
1563   </saml:AuthenticationStatement>
1564
1565   <!-- The Subject of the ResourceAccessStatement specifies use of
1566         bearer which indicates the subject of the statement
1567         is the sender of the message and is to be treated as the
1568         invocation identity for any access control decisions -->
1569   <ResourceAccessStatement xmlns="urn:liberty:sec:2003-08">
1570     <saml:Subject>
1571       <!-- the name identifier of the sender -->
1572       <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
1573         http://serviceprovider.com/
1574       </saml:NameIdentifier>
1575       <saml:SubjectConfirmation>
1576         <saml:ConfirmationMethod>
1577           urn:oasis:names:tc:SAML:1.0:cm:bearer
1578         </saml:ConfirmationMethod>
1579       </saml:SubjectConfirmation>
1580     </saml:Subject>
1581
1582     <!-- This ResourceID describes the resource for which
1583           the sender is attempting to access. -->
1584     <disco:ResourceID>http://example.com/disco/d0CQF8e1JTDLmzEo</disco:ResourceID>
1585   </ResourceAccessStatement>
1586   <!-- signature by the authority over the assertion -->
1587   <ds:Signature>...</ds:Signature>
1588 </saml:Assertion>
1589 <!-- this is the reference to the bearer token -->
1590 <wsse:SecurityTokenReference>
1591   <wsse:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
1592     ValueType="saml:Assertion" />
1593 </wsse:SecurityTokenReference>
1594 </wsse:Security>

```

```
1595 </s:Header>
1596 <s:Body wsu:Id="MsgBody">
1597   <pp:Modify>
1598     <!-- this is an ID-SIS-PP Modify message -->
1599   </pp:Modify>
1600 </s:Body>
1601 </s:Envelope>
1602
1603
```

1604 The next example depicts a custom security token being conveyed to the relying party. For such an example to function,
1605 the producer and consumer of the custom token must be able to determine the proper processing rules based off of the
1606 wsse:ValueType attribute.

```
1607 <?xml version="1.0" encoding="UTF-8"?>
1608 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1609   xmlns:sb="urn:liberty:sb:2003-08"
1610   xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1611   xmlns:sec="urn:liberty:sec:2003-08">
1612
1613   <s:Header>
1614     <sb:Correlation s:mustUnderstand="1"
1615       id="A13454...245"
1616       actor="http://schemas.../next"
1617       messageID="uuid:efefefef-aaaa-ffff-cc cc-eeeeffffbbbb"
1618       timestamp="2112-03-15T11:12:12Z"/>
1619
1620   <wsse:Security>
1621     <!-- Custom binary security token -->
1622     <wsse:BinarySecurityToken
1623       ValueType="anyNSPrefix:ServiceSessionContext"
1624       EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-sec
1625       urity-1.0#Base64Binary"
1626       wsu:Id="bst" >
1627       mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSrlv4
1628       YqUc70MiJcBtKbP3+jlD4HPUaurIqHA0vrDmMpM+sF2BnpND118f/mXCv3XbWhiL
1629       xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wS1TXNjCThHzBL8gBKZRqNbcZlU
1630       QXdp1/HIYQo5tIvCAM4pGk8nJFh6JrLsOEnT887aJRaasvBAAQ27C7D4Dmpt01aC
1631       FqLEQ98/lt6nkFmf7oiuZkID++xQXn74LWovdNlki43VaSXWcQAjzCzirHSuVX1N
1632       QvAsufa9Vghnry5Blxe2VzwtMDwiRCS/bpbRQAFEBQmR2FyeSBGLiBFbGxpc29u
1633       IDxnYXJ5LmVsbGlzB25Ac3VuLmNvbT6JARUDBRA0Z5icfpHfi79/fM0BARwaB/sG
1634       YHj+fpvMgRZev/i0DyZX+s6YyMZKeJ4pVHeboFP7KaP0R+VvAP0qoJk+6ITUYX2w
1635       R3eqeJPMbwqmoA/EAYkYE/xcqrq2ddSq2SG43530/TTOFY+ENxtt1tVhBdJ79KLx
1636       8fr2f9jLkjqQuB2MRKpy5EdJlqmtHkQm/SGTKRz8uncs5BtmJxkAbskuSi6Ys24E
1637       Pv0r97dW/uTfh7VM8+SA/hkCF6QVElUzvgpKwEpho2DZiuzvWAFqV/tINZRHGhCg
1638       TNlvyz+5yYXSAY3nr8UPzNJ9QUXrsmzB GDSlpqp3G07kL0VHN//B/5GLSVcofzpa
1639       xj/JP+41N4sDJGkyCWwqiQEeBBABAgAJBQI+d0xwAhkBAAoJEPCJEJL9ultFpMgH
1640       9AzI8pmuPKxv3dQcuqZ+rJRsy2YyuuSkWp j97n5PFWvBGTSau2+2wo3uLm8A596w
1641       n4MVShtx5SC2rMKKZABJ8ObqtbbSltQaIJmPg471qmnHjazeqbPfpPwQHzQ66cje
1642       De/3QbxBD/rPXV2SiyECed0qRsbuC90o3TonrJBOP6+Hs6jSkjGvQeJjvutukLMN
1643       A9T0d0CKN1rIEUwL4zweF7cmHWjWYfC64l8pqMFLC7XrYE7pXAL2Y6pi8Ta5njGL
1644       ldWryWzSDMCEunOt5wiuUYqZ+BXvy11kp2iKmi56ioTg5UHxGJqr6oZONDwMDIhw
1645       sI9v1kuHhJuWz8DziZ0li7QgR2FyeSBFbGxpc29uIDxnZmVAaW50ZXJoYWNrLm5l
1646       dD6JARQDBRA+d1WR8IkQkv26W0UBAXgsB/UROD8wayj9v7gMK3K9Idxk/3K16myl
1647       m0Q5mzFkXoLZ6EJ3wZlpxter9oeTo2F/5tJ0k9SFNaefFuipVGz9y+iDHHVKyQw
1648       kDGg7YB5+fK1siebpuIemvhngrUzLnmbOJDpBy+UukRGjRlHdSuEXN8fpGb27d
1649       ddo2odk31nr90pRPGo/F2mkduatD28MMPVn4RpOKw8Nx7PIIxVPnTXGgFLY2PDOO
1650       Dk5he7KszA3rJul9Dof0Ii9nLHlOXiHwXWFx71e66vw1HCIAwNpvU8BXSeIgbKDA
1651       ZzFMfUHsKyTdMo9l+ByDk/jLsGsvZ61tROShVWSw00rC8pKa3sVmsMy0C2dmZUBz
1652       dW4uY29tiQETAwQp3plwvCJEJL9ultFAQGRDgFwmhqrllACqYAr2a2yFoex0gIz
1653       NrTQvMjRWw5Eyz0Gu9KMQ5i1sBIpIHCa6LY/Y6rb0qsrP7Pu0Z082uuQAlfpRzs
1654       i4lHsZDOeKKAiw7G3bJO+fdpkwYPHC7YFObof45Y71BWO+OBfKkrMb73zfgYyGKIc
1655       tEcofKVO3fvNHNEdIEzhvY2o783JOGbdn34P5NcLre69eLPF3KNhonLQMVxlNmh
1656       0kw15rUckRPAPy4WgKv/VQEzXSPmx9t4x3jujcyDtsdvTnBMwEHUU3/Pn8TICa
1657       XsvFX/55u0PONTxFOi1A+0UpsCGrGpdzv1q7tRmF5FaOP1Um79Qg10/5060Gkdh
1658       cnkgRWxsaXNvbiA8Z2Z1QHN1bi5jb20+iQEUAwUQP3pmAvCJEJL9ultFAQF1twf0
```

```
1659     CAY7B8Nb74w+mYyHS+UXCrPQR21vs5DjzuKooX7j6pJHDQqhfss24NLBvvpufZa
1660     uTE27fDIx+HC0SK5cjGUTqoX/4nkMe+HM87vPcChbS3lTGT+yxVjyiQ9BIei5mX2
1661     QT19RkS3ZDXNux32uONDRX7dykNX6fYkKRGserWHhdXlHppmmvLodKCK/sZkkqzf
1662     VT4r9ytfpXBlue1OV93X8RUz4ecZcDm9e+IEG+pQjnvgrSgac1NrW5K/CJEOUjh
1663     oGTrym0Ziutezhrw/gOeLVtkywsMgDr77gWZxRvw01wl0gtUdTceuRBIDANj+KVZ
1664     vLk1TCaGAUNljkIDDgti
1665     =OuKj
1666     </wsse:BinarySecurityToken>
1667
1668     <!-- this is the reference to the above bearer token -->
1669     <wsse:SecurityTokenReference>
1670         <wsse:Reference URI="#bst" />
1671     </wsse:SecurityTokenReference>
1672 </wsse:Security>
1673 </s:Header>
1674 <s:Body wsu:Id="MsgBody">
1675     <!-- payload -->
1676 </s:Body>
1677 </s:Envelope>
1678
1679
```

1680 10. XSD

```
1681 <?xml version="1.0" encoding="UTF-8"?>
1682
1683 <xs:schema targetNamespace="urn:liberty:sec:2003-08"
1684     xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
1685     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1686     xmlns:ac="urn:liberty:ac:2003-08"
1687     xmlns:lib="urn:liberty:iff:2003-08"
1688     xmlns:disco="urn:liberty:disco:2003-08"
1689     xmlns:md="urn:liberty:metadata:2003-08"
1690     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1691     xmlns:sec="urn:liberty:sec:2003-08"
1692     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1693     elementFormDefault="qualified"
1694     attributeFormDefault="unqualified">
1695   <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
1696     schemaLocation="oasis-sstc-saml-schema-assertion-1.1.xsd"/>
1697   <xs:import namespace="urn:liberty:iff:2003-08"
1698     schemaLocation="liberty-idff-protocols-schema-1.2-errata-v3.0.xsd"/>
1699   <xs:import namespace="urn:liberty:disco:2003-08"
1700     schemaLocation="liberty-idwsf-disco-svc-v1.2.xsd"/>
1701   <xs:import namespace="urn:liberty:ac:2003-08"
1702     schemaLocation="liberty-authentication-context-v1.3.xsd"/>
1703   <xs:import namespace="urn:liberty:metadata:2003-08"
1704     schemaLocation="liberty-metadata-v1.1.xsd"/>
1705   <xs:import namespace="http://www.w3.org/2001/04/xmenc#"
1706     schemaLocation="http://www.w3.org/TR/2002/REC-xmenc-core-20021210/xenc-schema.xsd"/>
1707   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
1708     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-s
1709 chema.xsd"/>
1710
1711   <xs:annotation>
1712     <xs:documentation>Liberty ID-WSF Security Mechanisms Specification XSD</xs:documentation>
1713   </xs:annotation>
1714   The source code in this XSD file was excerpted verbatim from:
1715
1716   Liberty ID-WSF Security Mechanisms Specification
1717   Version 1.2
1718   14th December 2004
1719
1720   Copyright (c) 2003-2005 Liberty Alliance participants, see
1721   http://www.projectliberty.org/specs/idwsf_1_1_1_copyrights.php
1722
1723   </xs:documentation>
1724 </xs:annotation>
1725
1726   <xs:element name="ValidityRestrictionCondition" type="sec:ValidityRestrictionConditionType"/>
1727   <xs:complexType name="ValidityRestrictionConditionType">
1728     <xs:complexContent>
1729       <xs:extension base="saml:ConditionAbstractType">
1730         <xs:sequence>
1731           <xs:element name="NumberOfUses" type="xs:integer"/>
1732         </xs:sequence>
1733       </xs:extension>
1734     </xs:complexContent>
1735   </xs:complexType>
1736   <xs:element name="ProxySubject" substitutionGroup="saml:Subject"
1737     type="saml:SubjectType"/>
1738   <xs:annotation>
1739     <xs:documentation>ProxyTransitedStatement is a
1740     SubjectStatement which MAY carry specific subject confirmation
1741     data </xs:documentation>
1742   </xs:annotation>
1743   <xs:element name="ProxyTransitedStatement"
1744     type="saml:SubjectStatementAbstractType"/>
1745
```

```

1746     <xs:annotation>
1747     <xs:documentation>
1748         ProxyInfoConfirmationData may be relied upon to
1749         corroborate the path information carried in a
1750         ProxyTransitedStatement
1751 </xs:documentation>
1752 </xs:annotation>
1753 <xs:element name="ProxyInfoConfirmationData"
1754     type="sec:ProxyInfoConfirmationType"/>
1755 <xs:complexType name="ProxyInfoConfirmationType">
1756     <xs:sequence>
1757         <xs:element ref="saml:AssertionIDReference"/>
1758         <xs:element name="Issuer" type="xs:string"/>
1759         <xs:element name="IssueInstant" type="xs:dateTime"/>
1760         <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1"/>
1761     </xs:sequence>
1762     <xs:attribute name="id" type="xs:ID"/>
1763 </xs:complexType>
1764
1765 <xs:element name="SessionContext" type="sec:SessionContextType"/>
1766 <xs:complexType name="SessionContextType">
1767     <xs:sequence>
1768         <xs:element name="SessionSubject" type="lib:SubjectType"/>
1769         <xs:element name="ProviderID" type="md:entityIDType"/>
1770         <xs:element ref="lib:RequestAuthnContext" minOccurs="0" maxOccurs="1"/>
1771         <!-- The system entity for which this context applies
1772         is privacy protect by the SessionSubject -->
1773     </xs:sequence>
1774     <xs:attribute name="SessionIndex" type="xs:string" use="optional"/>
1775     <xs:attribute name="AuthenticationInstant" type="xs:dateTime" use="required"/>
1776     <xs:attribute name="AssertionIssueInstant" type="xs:dateTime" use="required"/>
1777 </xs:complexType>
1778
1779 <xs:element name="SessionContextStatement"
1780     type="sec:SessionContextStatementType"
1781     substitutionGroup="saml:SubjectStatement"/>
1782
1783 <xs:complexType name="SessionContextStatementType">
1784     <xs:complexContent>
1785         <xs:extension base="saml:SubjectStatementAbstractType">
1786             <xs:sequence>
1787                 <!-- This is the name of the proxy and it SHOULD carry
1788                 SubjectConfirmation information to authorize the
1789                 ProxySubject to act on behalf of the
1790                 Subject inherited from
1791                 SubjectStatementAbstractType -->
1792                 <xs:element name="ProxySubject"
1793                     type="saml:SubjectType" minOccurs="0"/>
1794                 <xs:element ref="sec:SessionContext"/>
1795             </xs:sequence>
1796         </xs:extension>
1797     </xs:complexContent>
1798 </xs:complexType>
1799
1800 <xs:element name="ResourceAccessStatement"
1801     type="sec:ResourceAccessStatementType"
1802     substitutionGroup="saml:SubjectStatement"/>
1803
1804 <xs:complexType name="ResourceAccessStatementType">
1805     <xs:complexContent>
1806         <xs:extension base="saml:SubjectStatementAbstractType">
1807             <xs:sequence>
1808                 <xs:group ref="disco:ResourceIDGroup"/>
1809                 <xs:sequence minOccurs="0">
1810                     <!-- This is the name of the proxy and it SHOULD carry
1811                     SubjectConfirmation information to authorize the
1812                     ProxySubject to act on behalf of the

```

```
1813         Subject inherited from
1814         SubjectStatementAbstractType -->
1815         <xs:element name="ProxySubject" type="saml:SubjectType"/>
1816         <xs:element ref="sec:SessionContext" minOccurs="0"/>
1817     </xs:sequence>
1818 </xs:sequence>
1819 </xs:extension>
1820 </xs:complexContent>
1821 </xs:complexType>
1822
1823 </xs:schema>
1824
```

1825 Bibliography

1826 Normative

- 1827 [LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 1.3, Liberty
1828 Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1829 [LibertyBindProf] Cantor, Scott, Kemp, John, Champagne, Darryl, eds. "Liberty ID-FF Bindings and
1830 Profiles Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004).
1831 <http://www.projectliberty.org/specs>
- 1832 [LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version
1833 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1834 [LibertyDisco] Sergent, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 1.2, Liberty
1835 Alliance Project (12 December 2004). <http://www.projectliberty.org/specs>
- 1836 [LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.1, Liberty
1837 Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1838 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. "Liberty ID-WSF SOAP Binding Specification
1839 ," Version 1.2, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1840 [LibertyGlossary] "Liberty Technical Glossary," Version 1.4, Liberty Alliance Project (14 Dec 2004).
1841 <http://www.projectliberty.org/specs> Hodges, Jeff, eds.
- 1842 [SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol
1843 for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification,
1844 version 1.1, Organization for the Advancement of Structured Information Standards [http://www.oasis-](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)
1845 [open.org/committees/documents.php?wg_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)
- 1846 [SAMLBind11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Bindings and Profiles
1847 for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification,
1848 version 1.1, Organization for the Advancement of Structured Information Standards [http://www.oasis-](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)
1849 [open.org/committees/documents.php?wg_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)
- 1850 [wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web
1851 Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for the
1852 Advancement of Structured Information Standards [http://docs.oasis-open.org/wss/2004/01/oasis-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
1853 [wss-soap-message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 1854 [wss-saml] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (December 1, 2004). Orga-
1855 nization for the Advancement of Structured Information Standards [http://docs.oasis-open.org/wss/oasis-wss-](http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf)
1856 [saml-token-profile-1.0.pdf](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
1857 200412],
- 1858 [wss-x509] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (March, 2004). Organi-
1859 zation for the Advancement of Structured Information Standards [http://docs.oasis-open.org/wss/2004/01/oasis-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)
1860 [200401-wss-x509-token-profile-1.0.pdf](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
1861 Standard V1.0 [OASIS 200401],
- 1862 [wss-kerb] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (May 5, 2003). Organi-
1863 zation for the Advancement of Structured Information Standards "Web Services Security: Kerberos Token
1864 Profile," Draft WSS-Kerberos-03,

-
- 1865 [XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and
1866 Processing," Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlsig-core>
- 1867 [xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (December 2002). "XML Encryption Syntax and Processing,"
1868 W3C Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlenc-core/>
- 1869 [RFC3268] Chown, P., eds. (June 2002). "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer
1870 Security (TLS)," RFC 3268., Internet Engineering Task Force <http://www.ietf.org/rfc/rfc3268.txt> [June
1871 2002].
- 1872 [SSL] Frier, A., Karlton, P., Kocher, P., eds. (November 1996). Netscape Communications Corporation "The SSL 3.0
1873 Protocol," <http://www.netscape.com/eng/ssl3/>
- 1874 [RFC2246] Dierks, T., Allen, C., eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet
1875 Engineering Task Force <http://www.ietf.org/rfc/rfc2246.txt> [January 1999].
- 1876 [Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May
1877 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium
1878 <http://www.w3.org/TR/xmlschema-1/>