



Liberty ID-WSF Security Mechanisms

Version: v2.0-03

Editors:

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

Contributors:

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

Abstract:

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

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

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 ActivCard; America Online, Inc.; American Express Travel Related Services; Axalto; Bank of
16 America Corporation; Bell Canada; Cingular Wireless; Cisco Systems, Inc.; Communicator, Inc.; Deloitte & Touche
17 LLP; Earthlink, Inc.; Electronic Data Systems, Inc.; Entrust, Inc.; Epok, Inc.; Ericsson; Fidelity Investments; France
18 Telecom; Gemplus; General Motors; Hewlett-Packard Company; i2 Technologies, Inc.; Internet2; Intuit Inc.;
19 MasterCard International; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nextel Communications; Nippon
20 Telegraph and Telephone Corporation; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OneName Corporation;
21 Openwave Systems Inc.; Phaos Technology; Ping Identity Corporation; PricewaterhouseCoopers LLP; RegistryPro,
22 Inc.; RSA Security Inc; Sabre Holdings Corporation; SAP AG; SchlumbergerSema; Sigaba; SK Telecom; Sony
23 Corporation; Sun Microsystems, Inc.; Symlabs, Inc.; Trustgenix; United Airlines; VeriSign, Inc.; Visa International;
24 Vodafone Group Plc; Wave Systems. 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 **Revision History**

- 32 **Revision:** 2.0-03 **Date:** 22 November 2004
33 Version, and schema update
34 **Revision:** 2.0-02 **Date:** 9 November 2004
35 Editorial
36 **Revision:** 2.0-01 **Date:** 4 October 2004
37 Overhaul for Samlv2.0

38 Contents

39	1. Abstract	5
40	2. Overview of Identity-Based Web Services Authorization (Informative)	6
41	3. Notation and Terminology	7
42	4. Security Requirements (Informative)	9
43	5. Confidentiality and Privacy Mechanisms	12
44	6. Authentication Mechanisms	14
45	7. Message Authorization Model	24
46	8. Supporting Schema	28
47	9. Examples (Informative)	29
48	10. Schema	43
49	Bibliography	44

50 1. Abstract

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

61 **2. Overview of Identity-Based Web Services Authorization (Infor-** 62 **mative)**

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

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

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

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

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

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

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

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

93 3. Notation and Terminology

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

97 3.1. Notational Conventions

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

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

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

105 3.2. Namespace

106 The following namespaces are referred to in this document:

107 **Table 1. Namespaces**

Prefix	Namespace
sec	<i>urn:liberty:sec:2004-12</i>
sb	<i>urn:liberty:sb:2003-08</i>
disco	<i>urn:liberty:disco:2003-08</i>
saml	<i>urn:oasis:names:tc:SAML:2.0: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

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

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

112 3.3. Terminology

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

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

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

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

122 **4. Security Requirements (Informative)**

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

126 **4.1. Security Requirements Overview**

127 There are multiple facets this security specification considers:

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

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

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

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

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

- 145 • Request Authentication
- 146 • Response Authentication
- 147 • Request/Response Correlation
- 148 • Replay Protection
- 149 • Integrity Protection
- 150 • Confidentiality Protection
- 151 • Privacy Protections
- 152 • Resource Access Authorization
- 153 • Proxy Authorization
- 154 • Mitigation of denial of service attack risks

155 **4.2. Common Requirements**

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

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

170 **4.3. Peer Authentication Requirements**

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

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

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

186 **4.4. Message Correlation Requirements**

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

189 **4.5. Privacy Requirements**

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

194 **4.6. Service Availability Requirements**

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

197 **4.7. Resource Access Authorization Requirements**

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

207 5. Confidentiality and Privacy Mechanisms

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

211 5.1. Transport Layer Channel Protection

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

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

219 • TLS_RSA_WITH_RC4_128_SHA

220 • TLS_RSA_WITH_3DES_EDE_CBC_SHA

221 • TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

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

226 • TLS_RSA_WITH_AES_CBC_SHA

227 • TLS_DHE_DSS_WITH_AES_CBC_SHA

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

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

232 **5.2. Message Confidentiality Protection**

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

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

241 **5.3. Identifier Privacy Protection**

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

247 **5.3.1. Encrypted Name Identifiers**

248 The identity conveyed in the subject MUST be resolvable in the namespace of the consuming service instance.
249 However, this requirement is in conflict with the need to protect the privacy of the identifier when the message passes
250 through intermediaries. To accomplish this securely the <saml:Subject> MUST contain a <saml:EncryptedID>
251 following the processing rules and recommendations specified in [\[SAMLCore2\]](#).

252 **5.3.2. Encrypted Attributes**

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

258 6. Authentication Mechanisms

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

- 262 • Peer Entity Authentication
- 263 • Message Authentication

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

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

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

272 **Table 2. Authentication Mechanisms**

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

273 6.1. Authentication Mechanism Overview (Informative)

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

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

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

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

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

- 292 • The X.509 v3 Certificate mechanism (Section 6.3.1) is suited for message exchanges that generally rely upon
293 message authentication as the principle factor in allowing the recipient to make authorization decisions.
- 294 • The SAML Assertion mechanism (Section 6.3.2) is suited for message exchanges that generally rely upon message
295 authentication as well as the conveyance and attestation of authorization information in order to allow the recipient
296 to make authorization decisions.
- 297 • The Bearer mechanism (Section 6.3.3) is based on the presence of a *bearer token* in the security header of a
298 message for which the sender does not explicitly demonstrate the right to lay claim to. In this case, the bearer
299 token is verified for authenticity rather than proving the authenticity of the message.

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

306 Not all peer entity authentication and message authentication combinations make sense in a given setting. Again this
307 is a matter of security policy and the trust model the policy accords. For example, in a conventional setting where
308 peer entity authentication is relied upon to ensure the authenticity, confidentiality and integrity of the transport in con-
309 junction with message authentication to assure message authorship, intent and retention of the act of attestation then
310 the mechanism *urn:liberty:security:2003-08:ClientTLS:X509* is relevant. However, such a combination may make
311 little sense when peer entity authentication is relied upon to imply message authentication. For example, the mecha-
312 nism *urn:liberty:security:2003-08:ClientTLS:X509* seems equivalent to *urn:liberty:security:2003-08:ClientTLS:null*
313 in such a setting. A similar argument can be made for the *urn:liberty:security:2004-12:ClientTLS:SAMLV2* mech-
314 anism. The relationship between the identity authenticated as a result of peer entity authentication and the identity
315 authenticated (or implied) from message authentication may diverge and describe two distinct system entities for ex-
316 ample, a system principal and a user principal respectively. The identities may also be required to reflect the same
317 system entities. This is a matter of deployment and operational policy and is out of scope for this specification.

318 6.2. Peer Entity Authentication

319 The Peer entity authentication mechanisms prescribed by this specification all rely upon the inherent security properties
320 of the TLS/SSL protocol (sometimes referred to as transport-level security); the different mechanisms differentiated by
321 how the message is authenticated. The mechanisms described below have distinct security properties regarding which
322 peers in a message exchange are authenticated. For the mechanisms that include both peer entity authentication and
323 message authentication, optimizations regarding attestation MAY be employed. For example, in environments where
324 there is no requirement that a signature attesting to the authenticity of the message be retained, then it may be sufficient
325 to rely upon the security properties of peer entity authentication to assure the integrity and authenticity of the message
326 payload with no additional message layer signature.

327 6.2.1. Unilateral Peer Entity Authentication

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

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

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

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

334 • *urn:liberty:security:2003-08:TLS:X509*

335 • *urn:liberty:security:2004-12:TLS:SAMLV2*

336 • *urn:liberty:security:2004-04:TLS:Bearer*

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

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

341 **6.2.1.1. 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.

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 **6.2.2. Mutual Peer Entity Authentication**

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

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

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

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

354 • *urn:liberty:security:2003-08:ClientTLS:X509*

355 • *urn:liberty:security:2004-12:ClientTLS:SAMLV2*

356 • *urn:liberty:security:2004-04:ClientTLS:Bearer*

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

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

361 **6.2.2.1. Processing Rules**

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

- 364 • The sender MUST authenticate the recipient AND the recipient MUST authenticate the sender.
- 365 • The recipient MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its
366 certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.
- 367 • The sender MUST authenticate using X.509 v3 certificates by demonstrating possession of the key bound to its
368 certificate in accordance with the processing rules and semantics of the TLS/SSL protocol.

369 **6.3. Message Authentication**

370 The non-null message authentication mechanisms prescribed by this specification generally rely upon the integrity
371 properties imbued by the application and verification of digital signatures over elements of the message header and
372 payload. The mechanisms described below have distinct security properties regarding authenticity of a given message.
373 For the mechanisms that include both peer entity authentication and message authentication, optimizations regarding
374 attestation MAY be employed. For example, in environments where there is no requirement that a signature attesting
375 to the authenticity of the message be retained, then it may be sufficient to rely upon the security properties of peer
376 entity authentication to assure the integrity and authenticity of the message payload with no additional message layer
377 signature.

378 **6.3.1. X.509 v3 Certificate Message Authentication**

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

- 380 • *urn:liberty:security:2003-08:null:X509*
- 381 • *urn:liberty:security:2003-08:TLS:X509*
- 382 • *urn:liberty:security:2003-08:ClientTLS:X509*

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

390 For deployment settings that require message authentication independent of peer entity authentication, then the sending
391 peer MUST perform message authentication by demonstrating proof of possession of a subject confirmation key
392 associated with the X.509 certificate. This key MUST be recognized by the recipient as belonging to the sending peer.

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

398 6.3.1.1. Sender Processing Rules

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

401 • The sender MUST demonstrate possession of a subject confirmation key.
402 For deployment settings which REQUIRE independent message authentication, the obligation MUST be accom-
403 plished by signing elements of the message and decorating the `<wsse:Security>` element with the signature.
404 For deployment settings which DO NOT REQUIRE independent message authentication then the sender MUST
405 accomplish this obligation by decorating the security header with a `<ds:KeyInfo>` element bearing the certificate.
406 This MUST be unambiguously verified to be the same certificate and key used in establishing peer entity
407 authentication. This is necessary to mitigate the threat of a certificate substitution attack. Also note that this
408 optimization only applies to the *urn:liberty:security:2003-08:ClientTLS:X509* mechanism.

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

411 • The `<sb:Correlation>` header block element.

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

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

415 • If the message is signed then the sender MUST include the resultant XML signature in a `<ds:Signature>`
416 element as a child of the `<wsse:Security>` header.

417 The `<ds:Signature>` element MUST refer to the subject confirmation key with a `<ds:KeyInfo>` element which
418 SHOULD carry a `<wsse:SecurityTokenReference>` element.

419 6.3.1.2. Recipient Processing Rules

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

-
- 422 • If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the
423 recipient MUST locate the <ds:KeyInfo> element bearing a security token. This token MUST be unambiguously
424 verified to be referring to the same certificate and key used in establishing peer entity authentication.
- 425 • If the message has been signed then the recipient MUST locate the <ds:Signature> element carried inside the
426 <wsse:Security> header.
427 The recipient MUST resolve the contents of the <ds:KeyInfo> element carried within the <ds:Signature>
428 and use the key it describes for validating the signed elements.
429 This validation MUST conform to the core validation rules described in [\[XMLDsig\]](#). Additionally, the recipient
430 MUST determine that it trusts the key used to sign the message, and the recipient SHOULD validate the sender's
431 certificate, verifying the certificate revocation status as appropriate to the risk of incorrect authentication.
- 432 • If peer entity authentication is not in use and the message is bound with [\[LibertySOAPBinding\]](#) the recipient
433 MUST verify the signature covers the following elements:
- 434 • The <sb:Correlation> header block element.
- 435 • All other header block elements that require the aforementioned security properties in accordance with the security
436 requirements prescribed in their respective specification.
- 437 • All sub-elements of the <S:Body>.

438 **6.3.2. SAML Assertion Message Authentication**

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

441 • *urn:liberty:security:2003-08:null:SAML*

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

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

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

446 • *urn:liberty:security:2004-12:null:SAMLV2*

447 • *urn:liberty:security:2004-12:TLS:SAMLV2*

448 • *urn:liberty:security:2004-12:ClientTLS:SAMLV2*

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

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

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

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

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

474 **6.3.2.1. Sender Processing Rules**

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

- 477 • The sender MUST present the <saml:Assertion> (as security token) by inserting it as a child of the
478 <wsse:Security> element.
- 479 • The sender MUST adhere to its subject confirmation obligation in accordance with the semantics of the
480 confirmation method described by one of the <saml:SubjectConfirmation> elements carried within the
481 <saml:Subject>.
- 482 For deployment settings which REQUIRE independent message authentication, the obligation MUST be accom-
483 plished by signing elements of the message and decorating the <wsse:Security> element with the signature.
484 For deployment settings which DO NOT REQUIRE independent message authentication then the subject confirma-
485 tion obligation may be accomplished by correlating the certificate and key used to affect peer entity authentication
486 with the certificate and key described by the subject confirmation element. To accommodate this, the assertion
487 issuing authority MUST construct the assertion such that the confirmation key can be unambiguously verified to
488 be the same certificate and key used in establishing peer entity authentication. This is necessary to mitigate the
489 threat of a certificate substitution attack. It is RECOMMENDED that the certificate or certificate chain be bound
490 to the subject confirmation key.
- 491 • If peer entity authentication is not used and the message is bound to SOAP with [\[LibertySOAPBinding\]](#) the sender
492 MUST sign:
- 493 • The <sb:Correlation> header block element.
- 494 • All other header block elements that require the aforementioned security properties in accordance with the security
495 requirements prescribed in their respective specification.
- 496 • All sub-elements of the <S:Body>.
- 497 • If the message is signed the sender MUST include the resultant XML signature in a <ds:Signature> element as
498 a child of the <wsse:Security> header
499 The <ds:Signature> element MUST refer to the subject confirmation key with a <ds:KeyInfo> element. The
500 <ds:KeyInfo> element SHOULD include a <wsse:SecurityTokenReference> element so that the subject
501 confirmation key can be located within the <wsse:Security> header. The inclusion of the reference SHOULD
502 adhere to the guidance specified in section 3.3.2 of [\[wss-saml\]](#).

503 6.3.2.2. Recipient Processing Rules

- 504 • The recipient MUST locate the <wsse:Security> element for which it is the target. This MUST adhere to the
505 rules specified in [\[wss-sms\]](#) and [\[wss-saml\]](#).
- 506 • The recipient MUST locate the <saml:Assertion> (security token) and the recipient MUST determine that it
507 trusts the authority which issued the <saml:Assertion>.
508 The recipient MUST validate the issuer's signature over the <saml:Assertion>. The recipient SHOULD
509 validate the trust semantics of the signing key, as appropriate to the risk of incorrect authentication.
- 510 • The recipient SHOULD verify that at least one of the confirmation obligations specified in the
511 <saml:SubjectConfirmation> element has been met.
- 512 • If the validation policy regards peer entity authentication sufficient for purposes of message authentication then the
513 recipient MUST locate the <ds:KeyInfo> element within <saml:SubjectConfirmation> element. This key
514 MUST be unambiguously verified to be referring to the same certificate and key used in establishing peer entity
515 authentication.

- 516 • If the message has been signed then the recipient MUST locate the `<ds:Signature>` element carried inside the
517 `<wsse:Security>` header.
518 The recipient MUST resolve the contents of the `<ds:KeyInfo>` element carried within the `<ds:Signature>`
519 and use the key it describes for validating the signed elements.
520 This validation MUST conform to the core validation rules described in [\[XMLDsig\]](#).
521 The recipient MUST determine that it trusts the key used to sign the message. The recipient SHOULD validate the
522 sender's certificate and verify the certificate revocation status, as appropriate to the risk of incorrect authentication.
- 523 • If peer entity authentication is not in use and the message is bound with [\[LibertySOAPBinding\]](#) the recipient
524 MUST verify the signature covers the following elements:
- 525 • The `<sb:Correlation>` header block element.
- 526 • All other header block elements that require the aforementioned security properties in accordance with the security
527 requirements prescribed in their respective specification.
- 528 • All sub-elements of the `<S:Body>`.

529 **6.3.3. Bearer Token Authentication**

530 The following URIs indicate bearer mechanisms:

- 531 • *urn:liberty:security:2004-04:null:Bearer*
- 532 • *urn:liberty:security:2004-04:TLS:Bearer*
- 533 • *urn:liberty:security:2004-04:ClientTLS:Bearer*

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

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

544 An example of a SAML bearer token can be found in [Section 9.5](#).

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

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

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

554 **6.3.3.1. Sender Processing Rules**

- 555 • The construction and decoration of the `<wsse:Security>` header element MUST adhere to the rules specified in
556 [\[wss-sms\]](#).
- 557 • The sender SHOULD wrap the bearer token within a `<wsse:Embedded>` element and make it a child of a
558 `<wsse:SecurityTokenReference>` as described in section 7.4 of [\[wss-sms\]](#).
559 The sender SHOULD indicate the type of the token by specifying the `wsse:ValueType` attribute of the
560 `<wsse:Embedded>` element.
- 561 • Alternatively the sender MAY simply insert the token within the `<wsse:Security>` header when the token type
562 is well known or takes an obvious form (e.g. `<wsse:BinarySecurityToken>`.)

563 **6.3.3.2. Recipient Processing Rules**

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

572 **7. Message Authorization Model**

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

579 **7.1. Authorization Mechanism Overview (Informative)**

580 The authorization mechanism defined by this specification formalizes the generation and conveyance of authorization
581 information. In support of this mechanism a Trusted Third Party (TTP) may be relied upon to act as either a Policy
582 Information Point (PIP), a Policy Decision Point (PDP) and potentially a coarse grained Policy Enforcement Point
583 (PEP). As a PIP the authority may facilitate the exchange of resource access information to the relying party. As
584 a PDP, the Trusted Third Party would adhere to the coarse access policies of the relying party insofar as ensuring
585 which entities may attempt to access a given resource. This requires strong assurance as to the authenticity of a peer
586 subject. Given the reliance of authorization upon authentication, this model aids in disseminating subject confirmation
587 obligations, identity information and access authorization data.

588 **7.2. Authorization Mechanism**

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

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

605 **7.3. Authorization Data Generation**

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

609 In addition to managing the registration and discovery of identity-based web services the trusted authority may act
610 as a centralized policy information and decision point. The authority may issue assertions regarding authentication
611 and authorization policies enforced for a given identity-based web service, resource and the identity of the sender.
612 The makeup of this assertion reflects the information necessary to accommodate the authentication and authorization
613 policy requirements.

614 **7.3.1. Processing Rules**

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

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

- 620 • The assertion **MUST** indicate the invocation identity within the `<saml:Subject>` element of the assertion.
621 The `<saml:Subject>` element **MUST** include at least one `<saml:SubjectConfirmation>` element. This ele-
622 ment **MUST** have a `Method` attribute with a value of `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key`.
623 The subject confirmation element **MUST** be specified with a `<saml:SubjectConfirmationData>` element
624 qualified with an `xsi:type` of `saml:KeyInfoConfirmationDataType` as specified in [SAMLCore2].
- 625 • When the invocation identity represents the identity of the sender, the `<saml:Subject>` element is decorated as
626 follows. Refer to [Section 9.1.1](#) for an informative example.
627 The name identifier element **SHOULD** include a `<saml:NameID>` element and the `Format` attribute value
628 **SHOULD** be `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note: This identifier might assist
629 the relying party in locating metadata concerning the subject of the assertion.
630 The `<saml:SubjectConfirmation>` element **SHOULD NOT** be decorated with a `<saml:NameID>` element.
- 631 • When the invocation identity is **NOT** that of the sender (i.e., the sender is acting as a proxy on behalf of the subject)
632 the `<saml:Subject>` element is decorated as follows:
633 In an operational setting where the invocation identity (the subject) is only to be released to the relying party
634 (the audience) then the name identifier element **SHOULD** be of type `<saml:EncryptedID>` and conform to the
635 guidance in [SAMLCore2]. Refer to [Section 9.1.2.2](#) for an informative example.
636 In settings where the invocation identity does not call for privacy protections then the name identifier element
637 **SHOULD** be conveyed using a `<saml:NameID>` element with a `Format` attribute which is appropriate for the
638 operational setting. Refer to [Section 9.1.2.1](#) for an informative example.
639 To identify the confirming entity the `<saml:SubjectConfirmation>` element **SHOULD** contain a
640 `<saml:NameID>` element with a `Format` attribute value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.
641 Note: This identifier might assist the relying party in locating metadata concerning the confirming entity as well
642 as help associate the name of the confirming entity in the application domain namespace with the key used for
643 subject confirmation.
- 644 • The assertion issuing authority **MAY** describe the authentication status of the interacting party by including
645 a `<saml:AuthnStatement>` element which **MUST** include a `<saml:AuthnContext>` element. Refer to
646 [Section 9.1.3](#) for an informative example.
- 647 • The assertion issuing authority **MAY** describe the resource for which sender intends to access at the relying party
648 by including an `<saml:AttributeStatement>`.
649 In an operational setting where the value of the attribute requires confidentiality protections then the attribute
650 element **SHOULD** be of type `<saml:EncryptedAttribute>` and conform to the guidance in [SAMLCore2].
651 Refer to [Section 9.1.4.2](#) for an informative example.
652 If the confidentiality of the attribute is not a concern then the element **SHOULD** be conveyed using a
653 `<saml:Attribute>`. Refer to [Section 9.1.4.1](#) for an informative example.
- 654 • **OPTIONALLY**, the assertion issuer **MAY** include information that assists in building a chain of transited proxies.
655 It is **RECOMMENDED** that the `<saml:Advice>` element be decorated with a `<saml:AssertionIDRef>` which
656 is a reference to the assertion bearing it. Also as the chain builds the assertion should be augmented with a
657 `<ProxyTransitedStatement>`. The issuer should include a `<saml:SubjectConfirmation>` for each proxy
658 (except for the last) that has participated in the progression of assertion issuance. See [Section 7.3.2](#) for a description
659 of how the proxy chain is constructed.

- 660 • The assertion MUST be signed by the assertion issuing authority in accordance with the signing requirements
661 specified in [\[SAMLCore2\]](#).

662 **7.3.2. Proxy Chaining**

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

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

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

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

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

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

682 See [Section 9.4](#) for an example of a `<saml:Assertion>` carrying a `<ProxyTransitedStatement>` with multiple
683 `<SubjectConfirmation>` elements.

684 7.4. Presenting Authorization Data

685 Interactions with identity-based web services may rely on the conveyance of authorization information. In general,
686 the a trusted authority issues the authorization data. In such a setting the authorization information would be sent
687 along with the identity-based web service request to the recipient. See [Authorization Data Generation \(Section 7.3\)](#)
688 for details as to how this data is acquired and formulated.

689 7.4.1. Processing Rules

- 690 • The sender **MUST** authenticate to the recipient using one of the authentication mechanisms described in [Message](#)
691 [Authentication \(Section 6.3\)](#).
692 It is **RECOMMENDED** that the sender authenticate using the [SAML Assertion Message Authentication](#) and
693 specifically conform to the processing rules specified in ([Section 6.3.2.1](#)).

694 7.5. Consuming Authorization Data

695 A recipient which exposes a resource typically makes access control decisions based on the invocation identity.
696 Additionally the recipient may also predicate access control policies upon the sender identity. The semantics of
697 resource access authorization are described in [Presenting Authorization Data \(Section 7.4\)](#).

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

704 7.5.1. Processing Rules

- 705 • The recipient **MUST** authenticate the sender using one of the mechanisms described in [Authentication Mechanisms](#)
706 ([Section 6.3.2](#)).
707 It is **RECOMMENDED** that the sender authenticate using the [SAML Assertion Message Authentication](#) and
708 specifically conform to the processing rules specified in ([Section 6.3.2.2](#)).
- 709 • The recipient **MUST** locate the `<saml:Assertion>` (security token) which conferred the subject confirmation
710 key relied upon for sender authentication.
711 The recipient **MUST** corroborate that the bound subject confirmation key is the same key used to authenticate the
712 communicating peer.
- 713 • The recipient **MUST** determine that it trusts the authority which signed the `<saml:Assertion>`.
714 The recipient **MUST** validate the signature of the `<saml:Assertion>`. The recipient **SHOULD** validate the trust
715 semantics of the signing key, as appropriate to the risk of incorrect authentication.

716 8. Supporting Schema

717 This section describes the additional schema elements that support the authorization model described in [Section 7](#).

718 8.1. ProxyTransitedStatement Schema

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

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

```
727
728 <xs:element name="ProxyTransitedStatement" type="sec:ProxyTransitedStatementType" />
729 <xs:complexType name="ProxyTransitedStatementType">
730 <xs:complexContent>
731 <xs:extension base="saml:StatementAbstractType">
732 <xs:sequence>
733 <xs:element ref="saml:SubjectConfirmation" minOccurs="1" maxOccurs="unbounded" />
734 </xs:sequence>
735 </xs:extension>
736 </xs:complexContent>
737 </xs:complexType>
738
```

739 8.2. ProxyInfoConfirmationData Schema

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

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

```
743 <xs:complexType name="ProxyInfoConfirmationDataType" mixed="false">
744 <xs:complexContent>
745 <xs:restriction base="saml:SubjectConfirmationDataType">
746 <xs:sequence>
747 <xs:element ref="saml:AssertionIDRef" />
748 <xs:element ref="saml:Issuer" />
749 <xs:element name="IssueInstant" type="xs:dateTime" />
750 <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1" />
751 </xs:sequence>
752 <xs:attribute name="id" type="xs:ID" />
753 </xs:restriction>
754 </xs:complexContent>
755 </xs:complexType>
756
```

757 The semantics around the elements are as follows:

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

763 9. Examples (Informative)

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

766 9.1. Fragmentary Examples

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

769 9.1.1. Sender as Invocation Identity

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

```
772
773 001 <saml:Subject>
774 002   <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
775 003     http://ovaloffice.whitehouse.gov/</saml:NameID>
776 004   <saml:SubjectConfirmation
777 005     Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
778 006     <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
779 007
780 008       <!-- This keyinfo is the key by which the sender must
781 009         prove possession in order for the relying party to
782 010         accept the Statements in this assertion. -->
783 011     <ds:KeyInfo>
784 012       <ds:KeyName>
785 013         CN=ovaloffice.whitehouse.gov,OU=Executive Branch,O=United States,...
786 014       </ds:KeyName>
787 015       <ds:KeyValue>...</ds:KeyValue>
788 016     </ds:KeyInfo>
789 017   </saml:SubjectConfirmationData>
790 018 </saml:SubjectConfirmation>
791 019 </saml:Subject>
```

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

796 9.1.2. Sender as Proxy Identity

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

801 9.1.2.1. Transparent Subject Identifier

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

```
804
805 001 <saml:Subject>
806 002   <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
807 003     president@whitehouse.gov</saml:NameID>
808 004   <saml:SubjectConfirmation
809 005     Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
810 006     <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
811 007       http://mailhost.whitehouse.gov/</saml:NameID>
```

```
812 008 <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
813 009
814 010 <!-- This keyinfo is the key by which the sender (aka proxy) must
815 011 prove possession in order for the relying party to
816 012 accept the Statements in this assertion. -->
817 013 <ds:KeyInfo>
818 014 <ds:KeyName>
819 015 CN=mailhost.whitehouse.gov,OU=Executive Branch,O=United States,...
820 016 </ds:KeyName>
821 017 <ds:KeyValue>...</ds:KeyValue>
822 018 </ds:KeyInfo>
823 019 </saml:SubjectConfirmationData>
824 020 </saml:SubjectConfirmation>
825 021 </saml:Subject>
826
```

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

831 9.1.2.2. Opaque Subject Identifier Identifier

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

```
834
835 001 <saml:Subject>
836 002 <saml:EncryptedID><xenc:EncryptedData>U2XTCNvRX7B11NK182nmY00TEk==</xenc:EncryptedData>
837 003 <xenc:EncryptedKey>...</xenc:EncryptedKey>
838 004 </saml:EncryptedID>
839 005 <saml:SubjectConfirmation
840 006 Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
841 007 <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
842 008 http://mailhost.whitehouse.gov/</saml:NameID>
843 009 <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
844 010
845 011 <!-- This keyinfo is the key by which the sender (aka proxy) must
846 012 prove possession in order for the relying party to
847 013 accept the Statements in this assertion. -->
848 014 <ds:KeyInfo>
849 015 <ds:KeyName>
850 016 CN=mailhost.whitehouse.gov,OU=Executive Branch,O=United States,...
851 017 </ds:KeyName>
852 018 <ds:KeyValue>...</ds:KeyValue>
853 019 </ds:KeyInfo>
854 020 </saml:SubjectConfirmationData>
855 021 </saml:SubjectConfirmation>
856 022 </saml:Subject>
857
```

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

860 9.1.3. Invoking Identity Authentication

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

```
863
864 001 <!-- The saml:AuthnStatement carries information that
865 002 describes the authentication event of the subject
866 003 to an authenticating authority -->
867 004 <saml:AuthnStatement
```

```
868 005 AuthnInstant="2005-04-01T16:57:30.000Z"
869 006 SessionIndex="6345789">
870 007 <saml:AuthnContext>
871 008   <saml:AuthnContextClassRef>
872 009     urn:oasis:names:tc:SAML:2.0:ac:classes>PasswordProtectedTransport
873 010   </saml:AuthnContextClassRef>
874 011 </saml:AuthnContext>
875 012 </saml:AuthnStatement>
876
```

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

880 9.1.4. Resource as an Attribute

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

886 9.1.4.1. Transparent Resource Identifier

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

```
888
889 001 <saml:AttributeStatement>
890 002   <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
891 003     Name="urn:liberty:disco:2005-04:ResourceID">
892 004     <saml:AttributeValue xsi:type="disco:ResourceID">
893 005       http://wsp.example.com/pp?id=foobar</saml:AttributeValue>
894 006   </saml:Attribute>
895 007 </saml:AttributeStatement>
896
```

897 9.1.4.2. Opaque Resource Identifier

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

```
901
902 001 <!-- The AttributeStatement carries an EncryptedAttribute.
903 002   Once this element is decrypted with the supplied key
904 003   an <Attribute> element bearing an <disco:ResourceID>
905 004   can be found. -->
906 005 <saml:AttributeStatement>
907 006   <saml:EncryptedAttribute>
908 007     <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
909 008       mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSrlv4
910 009       YqUc7OMiJcBtKBp3+jlD4HPUaurIqHA0vrDmMpM+sF2BnpND118f/mXCv3XbWhiL
911 010       xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZlU
912 011       ...
913 012       VRu9BpYBD4Y/98y1jtX9Pm898+xzketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
914 013       hg6nZ5c0I6L6Gn9A
915 014       =HCQY
916 015     </xenc:EncryptedData>
917 016     <xenc:EncryptedKey>
918 017       ...
919 018     </xenc:EncryptedKey>
920 019   </saml:EncryptedAttribute>
```

921 020 </saml:AttributeStatement>
 922

923 9.2. Proxying with Authentication Context of the Invoking Identity

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

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

930 The following example demonstrates:

- 931 • Proxying
- 932 • Encrypted Name Identifier
- 933 • Encrypted Resource Identifier

```

934
935 <?xml version="1.0" encoding="UTF-8"?>
936 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
937     xmlns:sb="urn:liberty:sb:2003-08"
938     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
939     xmlns:sec="urn:liberty:sec:2004-10"
940     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
941 rity-secext-1.0.xsd">
942
943
944 <s:Header>
945   <sb:Correlation s:mustUnderstand="1"
946     id="A13454...245"
947     actor="http://schemas.../next"
948     messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
949     timestamp="2112-03-15T11:12:12Z"/>
950 <wsse:Security>
951   <saml:Assertion
952     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
953     Version="2.0"
954     ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
955     IssueInstant="2005-04-01T16:58:33.173Z">
956
957     <saml:Issuer>http://authority.example.com/</saml:Issuer>
958
959     <!-- signature by the issuer over the assertion -->
960     <ds:Signature>...</ds:Signature>
961
962     <!-- By placing an audience restriction on the assertion we
963     can limit the scope of which entity should consume
964     the information in the assertion. -->
965
966     <saml:Conditions
967       NotBefore="2005-04-01T16:57:20Z"
968       NotOnOrAfter="2005-04-01T21:42:43Z">
969
970       <saml:AudienceRestrictionCondition>
971         <saml:Audience>http://wsp.example.com</saml:Audience>
972       </saml:AudienceRestrictionCondition>
973     </saml:Conditions>
974
975     <saml:Subject>
```



```

976     <saml:EncryptedID>
977       <xenc:EncryptedData>U2XTCNvRX7B1lNK182nmY00TEk==</xenc:EncryptedData>
978       <xenc:EncryptedKey>...</xenc:EncryptedKey>
979     </saml:EncryptedID>
980
981     <saml:SubjectConfirmation
982       Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
983       <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
984         http://wsc.example.com/</saml:NameID>
985       <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
986
987         <!-- This keyinfo is the key by which the sender must
988            prove possession in order for the relying party to
989            accept the Statements in this assertion. -->
990         <ds:KeyInfo>
991           <ds:KeyName>
992             CN=wsc.example.com,OU=Client Services R US,O=Service Station,...
993           </ds:KeyName>
994           <ds:KeyValue>...</ds:KeyValue>
995         </ds:KeyInfo>
996       </saml:SubjectConfirmationData>
997     </saml:SubjectConfirmation>
998 </saml:Subject>
999
1000 <!-- The AuthnStatement carries information
1001      that describes the authentication event
1002      of the Subject to an Authentication Authority -->
1003 <saml:AuthnStatement
1004   AuthnInstant="2005-04-01T16:57:30.000Z"
1005   SessionIndex="6345789">
1006   <saml:AuthnContext>
1007     <saml:AuthnContextClassRef>
1008       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
1009     </saml:AuthnContextClassRef>
1010   </saml:AuthnContext>
1011 </saml:AuthnStatement>
1012
1013 <!-- The AttributeStatement carries an EncryptedAttribute.
1014      Once this element is decrypted with the supplied key
1015      an <Attribute> element bearing an <disco:ResourceID>
1016      can be found. -->
1017 <saml:AttributeStatement>
1018   <saml:EncryptedAttribute>
1019     <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
1020       mQEMAZRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSrlv4
1021       YqUc70MiJcBtKBp3+jlD4HPUaurIqHA0vrmdMpm+sF2BnpND118f/mXCv3XbWhiL
1022       xjl/M4y0CMAM/wBHT3xa17tWJwsZkdRLWxXP7wSlTXNjCThHzBL8gBKZRqNBcZLU
1023       ...
1024       VRu9BpYBD4Y/98y1jtX9Pm898+xzketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
1025       hg6nZ5c0I6L6Gn9A
1026       =HCQY
1027     </xenc:EncryptedData>
1028     <xenc:EncryptedKey>...</xenc:EncryptedKey>
1029   </saml:EncryptedAttribute>
1030 </saml:AttributeStatement>
1031
1032 </saml:Assertion>
1033 <!-- this is the signature the sender generated to demonstrate holder-of-key
1034      the signature should cover the isf header and body-->
1035 <ds:Signature>
1036   <ds:SignedInfo>
1037     ...
1038     <ds:Reference URI="#A13454...245">
1039       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1040       <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1041     </ds:Reference>
1042     <ds:Reference URI="#MsgBody">

```

```

1043     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1044     <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1045     </ds:Reference>
1046   </ds:SignedInfo>
1047   <ds:KeyInfo>
1048     <wsse:SecurityTokenReference>
1049     <wsse:KeyIdentifier
1050       ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-1.
1051 0#SAMLAssertionID" />
1052       2sxJu9g/vvLG9sAN9bKp/8q0NKU=
1053     </wsse:KeyIdentifier>
1054   </ds:KeyInfo>
1055   <ds:SignatureValue>
1056     HJJWbvqW9E84vJVQkjJLLA6nNvBX7mY00TZhwBdFNDElgsCSXZ5Ekw==
1057   </ds:SignatureValue>
1058 </ds:Signature>
1059 </wsse:Security>
1060 </s:Header>
1061 <s:Body id="MsgBody">
1062   <pp:Modify>
1063     <!-- this is an ID-SIS-PP Modify message -->
1064   </pp:Modify>
1065 </s:Body>
1066 </s:Envelope>

```

1067 9.3. Conveyance of Sender as Invocation Identity

1068 This example depicts a request to access an identity-based web service in which the sender identity and the invocation
 1069 identity are the same (i.e. non-proxying). The resource which the sender is attempting to access is described in an
 1070 <AttributeStatement> within the assertion.

1071 Note that, while the assertion associates a subject's name with a key, this association is made as a means to indicate
 1072 the authorization of that subject, acting with that key, to invoke a service. This facility, incorporated for authorization
 1073 purposes, serves a distinct and complementary function to the binding between subject and key, which the subject's
 1074 certificate accomplishes for authentication purposes.

1075 The example demonstrates:

- 1076 • Sender is Invocation Identity.
- 1077 • Transparent Resource Identifier.

```

1078
1079 <?xml version="1.0" encoding="UTF-8"?>
1080 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1081   xmlns:sb="urn:liberty:sb:2003-08"
1082   xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1083   xmlns:sec="urn:liberty:sec:2004-10"
1084   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
1085 ity-secext-1.0.xsd">
1086
1087
1088   <s:Header>
1089     <sb:Correlation s:mustUnderstand="1"
1090       id="A13454...245"
1091       actor="http://schemas.../next"
1092       messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
1093       timestamp="2112-03-15T11:12:12Z"/>
1094   <wsse:Security>
1095     <saml:Assertion
1096       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1097       Version="2.0"
1098       ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="

```

```
1099 IssueInstant="2005-04-01T16:58:33.173Z">
1100
1101 <saml:Issuer>http://authority.example.com/</saml:Issuer>
1102
1103 <!-- signature by the issuer over the assertion -->
1104 <ds:Signature>...</ds:Signature>
1105
1106 <!-- By placing an audience restriction on the assertion we
1107 can limit the scope of which entity should consume
1108 the information in the assertion. -->
1109
1110 <saml:Conditions
1111   NotBefore="2005-04-01T16:57:20Z"
1112   NotOnOrAfter="2005-04-01T21:42:43Z">
1113
1114   <saml:AudienceRestrictionCondition>
1115     <saml:Audience>http://wsp.example.com</saml:Audience>
1116   </saml:AudienceRestrictionCondition>
1117 </saml:Conditions>
1118
1119 <saml:Subject>
1120   <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1121     http://ovaloffice.whitehouse.gov/</saml:NameID>
1122   <saml:SubjectConfirmation
1123     Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1124     <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
1125
1126       <!-- This keyinfo is the key by which the sender must
1127        prove possession in order for the relying party to
1128        accept the Statements in this assertion. -->
1129       <ds:KeyInfo>
1130         <ds:KeyName>
1131           CN=ovaloffice.whitehouse.gov,OU=Executive Branch,O=United States,...
1132         </ds:KeyName>
1133         <ds:KeyValue>...</ds:KeyValue>
1134       </ds:KeyInfo>
1135     </saml:SubjectConfirmationData>
1136   </saml:SubjectConfirmation>
1137 </saml:Subject>
1138
1139
1140 <saml:AttributeStatement>
1141   <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1142     Name="urn:liberty:disco:2005-04:ResourceID">
1143     <saml:AttributeValue xsi:type="disco:ResourceID">
1144       http://wsp.example.com/pp?id=foobar</saml:AttributeValue>
1145     </saml:Attribute>
1146   </saml:AttributeStatement>
1147 </saml:Assertion>
1148 <!-- this is the signature the sender generated to demonstrate holder-of-key
1149 the signature should cover the isf header and body-->
1150 <ds:Signature>
1151 <ds:SignedInfo>
1152   ...
1153   <ds:Reference URI="#A13454...245">
1154     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
1155     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1156   </ds:Reference>
1157   <ds:Reference URI="#MsgBody">
1158     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
1159     <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1160   </ds:Reference>
1161 </ds:SignedInfo>
1162 <ds:KeyInfo>
1163   <wsse:SecurityTokenReference>
1164   <wsse:KeyIdentifier
```

```

1166         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004
1167 XX-wss-saml-token-profile-1.0#SAMLAssertionID" />
1168         2sxJu9g/vvLG9sAN9bKp/8q0NKU=
1169         </wsse:KeyIdentifier>
1170     </ds:KeyInfo>
1171     <ds:SignatureValue>
1172         HJJWbvqW9E84vJVQkjJLLA6nNvBX7mY00TZhWbDFNDElgsCSXZ5Ekw==
1173     </ds:SignatureValue>
1174 </ds:Signature>
1175 </wsse:Security>
1176 </s:Header>
1177 <s:Body id="MsgBody">
1178     <pp:Modify>
1179         <!-- this is an ID-SIS-PP Modify message -->
1180     </pp:Modify>
1181 </s:Body>
1182 </s:Envelope>
1183
1184

```

1185 9.4. Proxy Chaining

1186 The following example demonstrates:

- 1187 • Proxy Chain captured in <ProxyTransitedStatement> as multiple <SubjectConfirmation> elements. Two
- 1188 different proxies separate from the sender are listed.
- 1189 • Encrypted Name Identifier.
- 1190 • Encrypted Resource Identifier.
- 1191 • Authentication status of Invoking Identity.

```

1192
1193 <?xml version="1.0" encoding="UTF-8"?>
1194 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1195     xmlns:sb="urn:liberty:sb:2004-04"
1196     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1197     xmlns:sec="urn:liberty:sec:2004-10"
1198     xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
1199 rity-secext-1.0.xsd">
1200
1201
1202 <s:Header>
1203     <sb:Correlation s:mustUnderstand="1"
1204         id="A13454...245"
1205         actor="http://schemas.../next"
1206         messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
1207         timestamp="2112-03-15T11:12:12Z"/>
1208 <wsse:Security>
1209     <saml:Assertion
1210         xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1211         Version="2.0"
1212         ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
1213         IssueInstant="2005-04-01T16:58:33.173Z">
1214
1215         <saml:Issuer>http://authority.example.com/</saml:Issuer>
1216
1217         <!-- signature by the issuer over the assertion -->
1218         <ds:Signature>...</ds:Signature>
1219
1220     <saml:Advice>
1221         <saml:AssertionIDRef>refers to this assertion</saml:AssertionIDReference>
1222     </saml:Assertion>

```

```

1223
1224     <!-- This statement reflects path of proxy transitions The
1225     list is comprised of SubjectConfirmation elements in the
1226     order the proxy was transitioned (first to last). -->
1227
1228     <sec:ProxyTransitedStatement>
1229         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1230             <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1231                 http://first.example.com/</saml:NameID>
1232             <saml:SubjectConfirmationData xsi:type="sec:ProxyInfoConfirmationData" type="a">
1233                 <saml:AssertionIDRef>
1234 <!-- refers to an assertion issued by the assertion issuer to first.example.com. -->
1235                 </saml:AssertionIDRef>
1236                 <saml:Issuer>authority.example.com</saml:Issuer>
1237                 <sec:IssueInstant>2004-04-01T16:58:30.173Z</sec:IssueInstant>
1238                 </saml:SubjectConfirmationData>
1239             </saml:SubjectConfirmation>
1240
1241             <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1242                 <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1243                     http://second.example.com/</saml:NameID>
1244                 <saml:SubjectConfirmationData xsi:type="sec:ProxyInfoConfirmationData" type="a">
1245                     <saml:AssertionIDRef>
1246 <!-- refers to an assertion issued by the assertion issuer to second.example.com. -->
1247                     </saml:AssertionIDRef>
1248                     <saml:Issuer>authority.example.com</saml:Issuer>
1249                     <sec:IssueInstant>2004-04-01T16:58:40.173Z</sec:IssueInstant>
1250                     </saml:SubjectConfirmationData>
1251                 </saml:SubjectConfirmation>
1252             </sec:ProxyTransitedStatement>
1253         </saml:Assertion>
1254     </saml:Advice>
1255
1256     <!-- By placing an audience restriction on the assertion we
1257     can limit the scope of which entity should consume
1258     the information in the assertion. -->
1259
1260     <saml:Conditions
1261         NotBefore="2005-04-01T16:57:20Z"
1262         NotOnOrAfter="2005-04-01T21:42:43Z">
1263
1264         <saml:AudienceRestrictionCondition>
1265             <saml:Audience>http://wsp.example.com</saml:Audience>
1266         </saml:AudienceRestrictionCondition>
1267     </saml:Conditions>
1268
1269     <saml:Subject>
1270         <saml:EncryptedID>
1271             <xenc:EncryptedData>U2XTCNvRX7B1lNK182nmY00TEk==</xenc:EncryptedData>
1272             <xenc:EncryptedKey>...</xenc:EncryptedKey>
1273         </saml:EncryptedID>
1274
1275         <saml:SubjectConfirmation
1276             Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1277             <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1278                 http://third.example.com/</saml:NameID>
1279             <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationData" type="a">
1280
1281                 <!-- This keyinfo is the key by which the sender must
1282                 prove possession in order for the relying party to
1283                 accept the Statements in this assertion. -->
1284                 <ds:KeyInfo>
1285                     <ds:KeyName>
1286                         CN=third.example.com,OU=Client Services R US,O=Service Station,...
1287                     </ds:KeyName>
1288                     <ds:KeyValue>...</ds:KeyValue>
1289                 </ds:KeyInfo>

```

```

1290     </saml:SubjectConfirmationData>
1291 </saml:SubjectConfirmation>
1292 </saml:Subject>
1293
1294 <!-- The AuthnStatement carries information
1295      that describes the authentication event
1296      of the Subject to an Authentication Authority -->
1297 <saml:AuthnStatement
1298   AuthnInstant="2005-04-01T16:57:30.000Z"
1299   SessionIndex="6345789">
1300   <saml:AuthnContext>
1301     <saml:AuthnContextClassRef>
1302       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
1303     </saml:AuthnContextClassRef>
1304   </saml:AuthnContext>
1305 </saml:AuthnStatement>
1306
1307 <!-- The AttributeStatement carries an EncryptedAttribute.
1308      Once this element is decrypted with the supplied key
1309      an <Attribute> element bearing an <disco:ResourceID>
1310      can be found. -->
1311 <saml:AttributeStatement>
1312   <saml:EncryptedAttribute>
1313     <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
1314       mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSrlv4
1315       YqUc70MiJcBtKbP3+jlD4HPUaurIqHA0vrdmMpm+sF2BnpND118f/mXCv3XbWhiL
1316       xjl/M4yOCMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCtHHzBL8gBKZRqNBcZlU
1317       ...
1318       VRu9BpYBD4Y/98y1jtX9Pm898+zxketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
1319       hg6nZ5c0I6L6Gn9A
1320       =HCQY
1321     </xenc:EncryptedData>
1322     <xenc:EncryptedKey> ... </xenc:EncryptedKey>
1323   </saml:EncryptedAttribute>
1324 </saml:AttributeStatement>
1325
1326 </saml:Assertion>
1327 <!-- this is the signature the sender generated to demonstrate holder-of-key
1328      the signature should cover the header and body-->
1329 <ds:Signature>
1330   <ds:SignedInfo>
1331     ...
1332     <ds:Reference URI="#A13454...245">
1333       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1334       <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1335     </ds:Reference>
1336     <ds:Reference URI="#MsgBody">
1337       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1338       <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1339     </ds:Reference>
1340   </ds:SignedInfo>
1341   <ds:KeyInfo>
1342     <wsse:SecurityTokenReference>
1343       <wsse:KeyIdentifier
1344         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-t
1345 oken-profile-1.0#SAMLAssertionID" />
1346         2sXJu9g/vvLG9sAN9bKp/8q0NKU=
1347       </wsse:KeyIdentifier>
1348     </ds:KeyInfo>
1349   </ds:SignatureValue>
1350     HJJWbvqW9E84vJVQkjJLLA6nNvBX7mY00TzhwBdFNDElgscSXZ5Ekw==
1351   </ds:SignatureValue>
1352 </ds:Signature>
1353 </wsse:Security>
1354 </s:Header>
1355 <s:Body id="MsgBody">
1356   <pp:Modify>

```

```

1357         <!-- this is an ID-SIS-PP Modify message -->
1358     </pp:Modify>
1359 </s:Body>
1360 </s:Envelope>
1361

```

1362 9.5. SAML Bearer Token

1363 The following example demonstrates the Bearer message authentication mechanism by supplying a SAML bearer
1364 token [wss-saml] in the security header.

```

1365
1366 <?xml version="1.0" encoding="UTF-8"?>
1367 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1368     xmlns:sb="urn:liberty:sb:2003-08"
1369     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1370     xmlns:sec="urn:liberty:sec:2004-10"
1371     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
1372 secext-1.0.xsd">
1373
1374     <s:Header>
1375         <sb:Correlation s:mustUnderstand="1"
1376             id="A13454...245"
1377             actor="http://schemas.../next"
1378             messageID="uuid:efefefef-aaaa-ffff-cccc-eeeefff fbbbb"
1379             timestamp="2112-03-15T11:12:12Z"/>
1380
1381         <wsse:Security>
1382             <!-- this is an embedded reference to the bearer token -->
1383             <wsse:SecurityTokenReference>
1384                 <wsse:Embedded
1385                     ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-p
1386 rofile-1.0#SAMLAssertionID">
1387
1388                 <saml:Assertion
1389                     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1390                     Version="2.0"
1391                     ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
1392                     IssueInstant="2005-04-01T16:58:33.173Z">
1393
1394                     <saml:Issuer>http://authority.example.com/</saml:Issuer>
1395
1396                     <!-- signature by the issuer over the assertion -->
1397                     <ds:Signature>...</ds:Signature>
1398
1399                     <!-- By placing an audience restriction on the assertion we
1400 can limit the scope of which entity should consume
1401 the information in the assertion. -->
1402
1403                     <saml:Conditions
1404                         NotBefore="2005-04-01T16:57:20Z"
1405                         NotOnOrAfter="2005-04-01T21:42:43Z">
1406
1407                         <saml:AudienceRestrictionCondition>
1408                             <saml:Audience>http://wsp.example.com</saml:Audience>
1409                         </saml:AudienceRestrictionCondition>
1410                     </saml:Conditions>
1411
1412                     <saml:Subject>
1413                         <saml:EncryptedID>
1414                             <xenc:EncryptedData>U2XTCNvRX7Bl1NK182 nmY00TEk==</xenc:EncryptedDat a>
1415                             <xenc:EncryptedKey>...</xenc:EncryptedKey>
1416                         </saml:EncryptedID>
1417
1418                         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm: bearer">
1419

```

```

1420     </saml:Subject>
1421
1422     <!-- The AuthnStatement carries information
1423           that describes the authentication event
1424           of the Subject to an Authentication Authority -->
1425     <saml:AuthnStatement
1426       AuthnInstant="2005-04-01T16:57:30.000Z"
1427       SessionIndex="6345789">
1428       <saml:AuthnContext>
1429         <saml:AuthnContextClassRef>
1430           urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
1431         </saml:AuthnContextClassRef>
1432       </saml:AuthnContext>
1433     </saml:AuthnStatement>
1434
1435     <!-- The AttributeStatement carries an EncryptedAttribute.
1436           Once this element is decrypted with the supplied key
1437           an <Attribute> element bearing an <disco:ResourceID>
1438           can be found. -->
1439     <saml:AttributeStatement>
1440       <saml:EncryptedAttribute>
1441         <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
1442           mQEMAzRniWkAAAEH9RWir0eKDkyFAB7P oFazx3ftp0vWwbbzqXdgcX8fpEqSr1v4
1443           YqUc7OMiJcBtKbP3+jlD4HPUaurIqHA0vrDmMpm+ sF2BnpND118f/mXCv3XbWhiL
1444           xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wS1TXNjCThh zBL8gBKZRqNBcZlU
1445           ...
1446           VRu9BpYBD4Y/98y1jtX9Pm898+zxketoc4ZvhCgh9P0arVK1B3cKxB87bKiDDWAU
1447           hg6nZ5c0I6L6Gn9A
1448           =HCQY
1449         </xenc:EncryptedData>
1450         <xenc:EncryptedKey> ... </xenc:EncryptedKey>
1451       </saml:EncryptedAttribute>
1452     </saml:AttributeStatement>
1453
1454   </saml:Assertion>
1455 </wsse:Embedded>
1456 </wsse:SecurityTokenReference>
1457 </wsse:Security>
1458 </s:Header>
1459 <s:Body id="MsgBody">
1460   <pp:Modify>
1461     <!-- this is an ID-SIS-PP Modify message -->
1462   </pp:Modify>
1463 </s:Body>
1464 </s:Envelope>
1465

```

1466 9.6. X.509 v3 Message Authentication

1467 The following example demonstrates X.509 v3 message authentication mechanism.

```

1468
1469 <?xml version="1.0" encoding="UTF-8"?>
1470   <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1471     xmlns:sb="urn:liberty:sb:2003-08"
1472     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1473     xmlns:sec="urn:liberty:sec:2003-08"
1474     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
1475 ssecurity-secext-1.0.xsd">
1476
1477   <s:Header>
1478     <sb:Correlation s:mustUnderstand="1"
1479       id="A13454...245"
1480       actor="http://schemas.../next"
1481       messageID="uuid:efefefef-aaaa-ff ff-cccc-eeeeffffbbbb"

```



```

1483         timestamp="2112-03-15T11:12:12Z"/>
1484     <wsse:Security xmlns:wsse="...">
1485         <wsse:BinarySecurityToken ValueType="wsse:X509v3" wsu:Id="X509Token"
1486             EncodingType="wsse:Base64Binary">
1487             MIIB9zCCAWSgAwIBAgIQ...
1488         </wsse:BinarySecurityToken>
1489         <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1490             <ds:SignedInfo>
1491
1492                 <!-- bind the correlation header -->
1493                 <ds:Reference URI="#A13454...245">
1494                     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1495                     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
1496                 </ds:Reference>
1497                 <!-- bind the security token (thwart cert substitution attacks) -->
1498                 <ds:Reference URI="#X509Token">
1499                     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1500                     <ds:DigestValue>Ru4cAfeBABE...</ds:DigestValue>
1501                 </ds:Reference>
1502                 <!-- bind the body of the message -->
1503                 <ds:Reference URI="#MsgBody">
1504                     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1505                     <ds:DigestValue>YgGfS0pi56pu...</ds:DigestValue>
1506                 </ds:Reference>
1507             </ds:SignedInfo>
1508             <ds:KeyInfo>
1509                 <wsse:SecurityTokenReference>
1510                     <wsse:Reference URI="#X509Token" />
1511                 </wsse:SecurityTokenReference>
1512             </ds:KeyInfo>
1513             <ds:SignatureValue>
1514                 HJJWbvqW9E84vJVQkjJLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw==
1515             </ds:SignatureValue>
1516         </ds:Signature>
1517     </wsse:Security>
1518 </s:Header>
1519 <s:Body id="MsgBody">
1520     <pp:Modify>
1521         <!-- this is an ID-SIS-PP Modify message -->
1522     </pp:Modify>
1523 </s:Body>
1524 </s:Envelope>
1525

```

1526 9.7. Custom Bearer Token Message Authentication

1527 This example depicts a custom security token being conveyed to the relying party. For such an example to function,
1528 the producer and consumer of the custom token must be able to determine the proper processing rules based off of the
1529 wsse:ValueType attribute.

```

1530
1531 <?xml version="1.0" encoding="UTF-8"?>
1532 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
1533     xmlns:sb="urn:liberty:sb:2003-08"
1534     xmlns:pp="urn:liberty:id-sis-pp:2003-08"
1535     xmlns:sec="urn:liberty:sec:2004-10"
1536     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
1537     secext-1.0.xsd">
1538
1539     <s:Header>
1540         <sb:Correlation s:mustUnderstand="1"
1541             id="A13454...245"
1542             actor="http://schemas.../next"
1543             messageID="uuid:efefefef-aaaa-ffff-cccc-eeeefffffbbbb"
1544             timestamp="2112-03-15T11:12:12Z"/>
1545

```

```
1546 <wsse:Security>
1547 <!-- Custom binary security token -->
1548 <wsse:BinarySecurityToken
1549   ValueType="anyNSPrefix:ServiceSessionContext"
1550   EncodingType="wsse:Base64Binary"
1551   wsu:Id="bst" />
1552 mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8fpEqSrlv4
1553 YqUc70MiJcBtKBp3+jlD4HPUaurIqHA0vrdmMpm+sF2BnpND118f/mXCv3XbWhiL
1554 xj1/M4y0CMAM/wBHT3xa17tWJwsZkDRLWxXP7wSlTXNjCThHzBL8gBKZRqNbcZlU
1555 QXdp1/HIYQo5tIvCAM4pGk8nJFh6JrLsOEnT887aJRaasvBAAQ27C7D4Dmpt01aC
1556 FqLEQ98/lt6nkFmf7oiuZkID++xQXn74LW0vdNlki43VaSXWcQAjzCzirHSuVX1N
1557 QvAsufa9Vghnry5Blxe2VzwtMDwiRCS/bpbRQAFebQmR2FyeSBGLiBFbGxpc29u
1558 IDxnYXJ5LmVsbG1zb25Ac3VuLmNvbT6JARUDBRA0Z5icfpHfi79/fM0BARwaB/sG
1559 YHj+fpvMgRZev/i0DyZX+s6YyMZKeJ4pVHeboFP7KaP0R+VvAP0qoJK+6ITUyX2w
1560 R3eqeJPMbWqmOA/EAYkYE/xcqrq2ddSg2SG43530/TTOFY+ENXtltVhBdJ79KLx
1561 8fr2f9jLKJqQB2MRKpy5EdJlqmtHkQm/SGTKRz8uncs5BtmJxkAbskuSi6Ys24E
1562 Pv0r97dW/uTfh7VM8+SA/hkCF6QVElUzvgpKwEph2DZiuzvWAFqV/tINZRHGhCg
1563 TNlvyz+5yYXSAY3nr8UPzNJ9QUXrsmzBGDSlppq3GO7kL0VHN//B/5GLSVcofzpa
1564 xj/JP+41N4sDJGkyCWwqiQEeBBABAgAJBQI+d0xwAhkBAOJEPCEJEL9ultFpMgH
1565 9AzI8pmuPKxv3QcucqZ+rJRsy2YyuuSkWpJ97n5PFWvBGTSAu2+2wo3uLm8A596w
1566 n4MVShtx5SC2rMKKZABJ8ObqtbbSltQaIJmPg471qmnHjazeqbPfpPwPQHzQ66cje
1567 De/3QbxBd/rPXV2SiyECed0qRsbuC90o3TonrJBop6+Hs6jSkjGvQeJjvutukLMN
1568 A9T0d0CKN1riEUWl4zweF7cmHWjWYfC64l8pqMFLC7XrYE7pXAL2Y6pi8Ta5njGL
1569 ldWryWzSDMCEunOt5wiuUYqZ+BXvy1lkp2iKmi56ioTg5UHxGJqr6oZONDWMDIhW
1570 sI9v1kuHhJuWz8DZiZ0Ii7QgR2FyeSBFBGxpc29uIDxnZmVAaW50ZXJoYWNrLm5l
1571 dD6JARQDBRA+d1WR8IkQkv26W0UBAXgsB/UROD8wayj9v7gMK3K9Idxk/3K16myl
1572 m0Q5mzFkXoLZ6EJ3wZlpxter9oeTo2F/5tJ0k9SFNaefFuiPVGz9y+iDHHVKyQw
1573 kDGG7YB5+fK1siebPUnIemvhngrUzLnmbOJDpBy+UukRGjRlhdSuEXN8fpGb27d
1574 ddo2odK3lnR9oPRPGo/F2mkduatD28MMPVn4RpOKw8Nx7PIIxVpNTXGgfLY2PDOO
1575 Dk5he7KszA3rJul9Dof0Ii9nLHlOXiHwXWfx71e66vwLHC IaNwPvU8BXSeIgbKDA
1576 ZzFMfUHsKyTdMo9l+ByDk/jLsGsvZ61tROS hVWSw0rC8pKa3sVmSMY0C2dmZUBz
1577 dW4uY29tiQETAwUQP3plwvCJEJL9ultFAQGRDgfwmhqrrlACqYAr2a2yFoex0gIz
1578 NrTQvMJRWw5Eyz0Gu9KMQ5i1sBIpIHCCa6LY/Y6rb0qsrP7Pu0Z082uuQAlfpRzs
1579 i4lHsZDOeKKAiw7G3bJO+fDpkwYPHC7YFObof45Y71BWO+OBfKkrMb73zfgYYGK Ic
1580 tECofkVO3fvNHNEdIEzhvY2o783JOGbdN34P5NcLre69eLPF3K NhonLQMVx1Nmh
1581 0kwl5rUckRPAPy4WgKv/VQEztXSPmx9t4x3jUjcyDtSdvTnBMwEHUU3/Pn8TICa
1582 XsvFX/55u0PontXfoiLA+0UpsCGrGpdzvlq7tRmF5aOP1Um79Qg10/5060Gkdh
1583 cnkgRWxsaXNvbiA8Z2Z1QHNLbi5jB20+iQEUAWUQP3pmAvCJEJL9ultFAQF1twf0
1584 CAY7B8Nb74w+mYYyHS+UXCrPQR21vs5DjzukoX7j6pJHDQqhfss24NLBvvpufZa
1585 uTE27fDIx+HC0SK5cJGUTqoX/4nkMe+HM87vPcChbS31TGT+yxVjyiQ9BIei5mX2
1586 QT19Rks3ZDXNux32uONDRX7dykNX6fYkKRgserWHhdX1HppmmvLodKCK/sZkkqzf
1587 VT4r9ytfpXBlue1OV93XRuZ4ecZcDm9e+IEG+pQjnvgrSgac1NrW5K/CJEOUJh
1588 oGTrym0Ziutezhwr/goeLVtkywsMgDr77gWZxRvw01wl0gtUdTceuRBIDANj+KVZ
1589 vLKLtCaGAUNIJkiDDgti
1590 =OuKj
1591 </wsse:BinarySecurityToken>
1592
1593 <!-- this is the reference to the above bearer token -->
1594 <wsse:SecurityTokenReference>
1595   <wsse:Reference URI="#bst" />
1596 </wsse:SecurityTokenReference>
1597 </wsse:Security>
1598 </s:Header>
1599 <s:Body id="MsgBody">
1600   <!-- payload -->
1601 </s:Body>
1602 </s:Envelope>
1603
```

1604 10. Schema

```

1605 <?xml version="1.0" encoding="UTF-8"?>
1606
1607 <xs:schema targetNamespace="urn:liberty:sec:2004-12"
1608     xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
1609     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1610     xmlns:disco="urn:liberty:disco:2004-12"
1611     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1612     xmlns:sec="urn:liberty:sec:2004-12"
1613     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1614     xmlns:md="urn:liberty:metadata:2004-12"
1615     elementFormDefault="qualified"
1616     attributeFormDefault="unqualified">
1617     <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1618         schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
1619     <xs:import namespace="urn:liberty:disco:2004-12"
1620         schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
1621     <!-- <xs:import namespace="urn:liberty:ac:2004-12"
1622         schemaLocation="liberty-authentication-context-v2.0.xsd"/> -->
1623     <xs:import namespace="urn:liberty:metadata:2004-12"
1624         schemaLocation="liberty-metadata-v2.0.xsd"/>
1625     <xs:import namespace="http://www.w3.org/2001/04/xmenc#"
1626         schemaLocation="http://www.w3.org/TR/2002/REC-xmenc-core-20021210/xenc-schema.xsd"/>
1627     <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
1628         schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-co
1629 re-schema.xsd"/>
1630     <xs:annotation>
1631         <xs:documentation>Liberty ID-WSF Security Mechanisms Specification XSD</xs:documentation>
1632     </xs:annotation>
1633     The source code in this XSD file was excerpted verbatim from:
1634
1635     Liberty ID-WSF Security Mechanisms Specification
1636     Version 2.0-03
1637     22 November 2004
1638
1639         Copyright (c) 2004 Liberty Alliance participants, see
1640         http://www.projectliberty.org/specs/idwsf_2_0_copyrights.php
1641
1642     </xs:documentation>
1643 </xs:annotation>
1644
1645 <xs:element name="ProxyTransitedStatement" type="sec:ProxyTransitedStatementType"/>
1646 <xs:complexType name="ProxyTransitedStatementType">
1647     <xs:complexContent>
1648         <xs:extension base="saml:StatementAbstractType">
1649             <xs:sequence>
1650                 <xs:element ref="saml:SubjectConfirmation" minOccurs="1" maxOccurs="unbounded"/>
1651             </xs:sequence>
1652         </xs:extension>
1653     </xs:complexContent>
1654 </xs:complexType> <xs:complexType name="ProxyInfoConfirmationDataType" mixed="false">
1655     <xs:complexContent>
1656         <xs:restriction base="saml:SubjectConfirmationDataType">
1657             <xs:sequence>
1658                 <xs:element ref="saml:AssertionIDRef"/>
1659                 <xs:element ref="saml:Issuer" />
1660                 <xs:element name="IssueInstant" type="xs:dateTime"/>
1661                 <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1"/>
1662             </xs:sequence>
1663             <xs:attribute name="id" type="xs:ID"/>
1664         </xs:restriction>
1665     </xs:complexContent>
1666 </xs:complexType>
1667 </xs:schema>

```

Bibliography

Normative References

- 1668
1669
- 1670 [LibertySecMech] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.1, Liberty Alliance Project
1671 (18 April 2004). <http://www.projectliberty.org/specs>
- 1672 [LibertySecMechV20] Ellison, Gary, Madsen, Paul, eds. "Liberty ID-WSF Security Mechanisms," Version 2.0-03,
1673 Liberty Alliance Project (22 November 2004). <http://www.projectliberty.org/specs>
- 1674 [LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 2.0-01,
1675 Liberty Alliance Project (21 November 2004). <http://www.projectliberty.org/specs>
- 1676 [LibertyBindProf] Cantor, Scott, Kemp, John, Champagne, Darryl, eds. "Liberty ID-FF Bindings and
1677 Profiles Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004).
1678 <http://www.projectliberty.org/specs>
- 1679 [LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version
1680 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). <http://www.projectliberty.org/specs>
- 1681 [LibertyDisco] Beatty, John, Hodges, Jeff, Sergent, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification,"
1682 Version 2.0-02, Liberty Alliance Project (24 Nov 2004). <http://www.projectliberty.org/specs>
- 1683 [LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 2.0-02,
1684 Liberty Alliance Project (25 November 2004). <http://www.projectliberty.org/specs>
- 1685 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. "Liberty ID-WSF SOAP Binding Specification
1686 ," Version 2.0-01, Liberty Alliance Project (22 November 2004). <http://www.projectliberty.org/specs>
- 1687 [LibertyGlossary] Hodges, Jeff, eds. "Liberty Technical Glossary," Version 1.3-errata-v1.0, Liberty Alliance Project
1688 (12 Aug 2004). <http://www.projectliberty.org/specs>
- 1689 [SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol
1690 for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification,
1691 version 1.1, Organization for the Advancement of Structured Information Standards [http://www.oasis-
open.org/committees/documents.php?wg_abbrev=security](http://www.oasis-
1692 open.org/committees/documents.php?wg_abbrev=security)
- 1693 [SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (24 September 2004). "Assertions
1694 and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," OASIS Committee
1695 Draft 02, v2.0 , Organization for the Advancement of Structured Information Standards [http://www.oasis-
open.org/committees/security/](http://www.oasis-
1696 open.org/committees/security/)
- 1697 [SAMLBind11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Bindings and Profiles
1698 for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification,
1699 version 1.1, Organization for the Advancement of Structured Information Standards [http://www.oasis-
open.org/committees/documents.php?wg_abbrev=security](http://www.oasis-
1700 open.org/committees/documents.php?wg_abbrev=security)
- 1701 [SAMLBind2] Hughes, John, Cantor, Scott, Mishra, Prateek, Hirsch, Frederick, Philpott, Rob, Hodges, Jeff, Maler,
1702 Eve, eds. (18 August 2004). "Bindings and Profiles for the OASIS Security Assertion Markup Language
1703 (SAML) V2.0," OASIS Committee Specification, version 2.0, Organization for the Advancement of Struct-
1704 tured Information Standards http://www.oasis-open.org/committees/documents.php?wg_abbrev=security
- 1705 [wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web
1706 Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for the
1707 Advancement of Structured Information Standards [http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1708 wss-soap-message-security-1.0.pdf)

- 1709 [wss-saml] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (May 5,
1710 2003). Organization for the Advancement of Structured Information Standards [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1911/WSS-SAML-07.pdf)
1711 [open.org/committees/download.php/1911/WSS-SAML-07.pdf](http://www.oasis-open.org/committees/download.php/1911/WSS-SAML-07.pdf) "Web Services Security: SAML Token
1712 Profile," Draft WSS-SAML-07.pdf,
- 1713 [wss-x509] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (March, 2004). Organiza-
1714 tion 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)
1715 [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
1716 Standard V1.0 [OASIS 200401],
- 1717 [wss-kerb] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (May 5, 2003). Organi-
1718 zation for the Advancement of Structured Information Standards "Web Services Security: Kerberos Token
1719 Profile," Draft WSS-Kerberos-03,
- 1720 [XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and
1721 Processing," Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlsig-core>
- 1722 [xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (December 2002). "XML Encryption Syntax and Processing,"
1723 W3C Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlenc-core/>
- 1724 [RFC3268] Chown, P., eds. (June 2002). "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer
1725 Security (TLS)," RFC 3268., Internet Engineering Task Force <http://www.ietf.org/rfc/rfc3268.txt> [June
1726 2002].
- 1727 [SSL] Frier, A., Karlton, P., Kocher, P., eds. (November 1996). Netscape Communications Corporation "The SSL 3.0
1728 Protocol," <http://www.netscape.com/eng/ssl3/>
- 1729 [RFC2246] Dierks, T., Allen, C., eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet
1730 Engineering Task Force <http://www.ietf.org/rfc/rfc2246.txt> [January 1999].
- 1731 [Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May
1732 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium
1733 <http://www.w3.org/TR/xmlschema-1/>