



Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

Committee Draft 02, 24 September 2004

Document identifier:

sstc-saml-core-2.0-cd-02

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:

Scott Cantor, Internet2
John Kemp, Nokia
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems

SAML V2.0 Contributors:

Conor P. Cahill, AOL
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rick Randall, Booze, Allen, Hamilton
Tim Alsop, CyberSafe Limited
Nick Ragouzis, Enosis
John Hughes, Atos Origin
Paul Madsen, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Rebekah Metz, NASA
Prateek Mishra, Netegrity
Peter C Davis, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems

42 Jeff Hodges, Sun Microsystems
43 Eve Maler, Sun Microsystems
44 Ron Monzillo, Sun Microsystems
45 Greg Whitehead, Trustgenix

46 **Abstract:**

47 This specification defines the syntax and semantics for XML-encoded assertions about
48 authentication, attributes, and authorization, and for the protocols that convey this information.

49 **Status:**

50 This is a **second Committee Draft** approved by the Security Services Technical Committee on
51 21 September 2004.

52 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
53 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by filling out the web form located
54 at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security. The
55 committee will publish on its web page (<http://www.oasis-open.org/committees/security>) a catalog
56 of any changes made to this document as a result of comments.

57 For information on whether any patents have been disclosed that may be essential to
58 implementing this specification, and any offers of patent licensing terms, please refer to the
59 Intellectual Property Rights web page for the Security Services TC ([http://www.oasis-](http://www.oasis-open.org/committees/security/ipr.php)
60 [open.org/committees/security/ipr.php](http://www.oasis-open.org/committees/security/ipr.php)).

61 Table of Contents

62	1 Introduction.....	7
63	1.1 Notation.....	7
64	1.2 Schema Organization and Namespaces.....	8
65	1.2.1 String Values.....	8
66	1.2.2 URI Values.....	9
67	1.2.3 Time Values.....	9
68	1.2.4 ID and ID Reference Values.....	9
69	2 SAML Assertions.....	11
70	2.1 Schema Header and Namespace Declarations.....	11
71	2.2 Name Identifiers.....	12
72	2.2.1 Element <BaseID>.....	12
73	2.2.2 Element <NameID>.....	13
74	2.2.3 Element <EncryptedID>.....	14
75	2.2.4 Element <Issuer>.....	14
76	2.3 Assertions.....	14
77	2.3.1 Element <AssertionIDRef>.....	14
78	2.3.2 Element <AssertionURIRef>.....	15
79	2.3.3 Element <Assertion>.....	15
80	2.3.4 Element <EncryptedAssertion>.....	16
81	2.4 Subjects.....	17
82	2.4.1 Element <Subject>.....	17
83	2.4.1.1 Element <SubjectConfirmation>.....	18
84	2.4.1.1.1 Element <SubjectConfirmationData>.....	18
85	2.4.1.2 Complex Type KeyInfoConfirmationDataType.....	20
86	2.5 Conditions.....	20
87	2.5.1 Element <Conditions>.....	20
88	2.5.1.1 Attributes NotBefore and NotOnOrAfter.....	22
89	2.5.1.2 Element <Condition>.....	22
90	2.5.1.3 Elements <AudienceRestriction> and <Audience>.....	22
91	2.5.1.4 Element <OneTimeUse>.....	23
92	2.5.1.5 Element <ProxyRestriction>.....	24
93	2.6 Advice.....	24
94	2.6.1 Element <Advice>.....	25
95	2.7 Statements.....	25
96	2.7.1 Element <Statement>.....	25
97	2.7.2 Element <AuthnStatement>.....	25
98	2.7.2.1 Element <SubjectLocality>.....	27
99	2.7.2.2 Element <AuthnContext>.....	27
100	2.7.3 Element <AttributeStatement>.....	28
101	2.7.3.1 Element <Attribute>.....	28
102	2.7.3.1.1 Element <AttributeValue>.....	29
103	2.7.3.2 Element <EncryptedAttribute>.....	30
104	2.7.4 Element <AuthzDecisionStatement>.....	30
105	2.7.4.1 Simple Type DecisionType.....	32
106	2.7.4.2 Element <Action>.....	32
107	2.7.4.3 Element <Evidence>.....	32
108	3 SAML Protocols.....	34
109	3.1 Schema Header and Namespace Declarations.....	34
110	3.2 Requests and Responses.....	35

111	3.2.1 Complex Type RequestAbstractType.....	35
112	3.2.2 Complex Type StatusResponseType.....	36
113	3.2.2.1 Element <Status>.....	38
114	3.2.2.2 Element <StatusCode>.....	38
115	3.2.2.3 Element <StatusMessage>.....	40
116	3.2.2.4 Element <StatusDetail>.....	41
117	3.3 Assertion Query and Request Protocol.....	41
118	3.3.1 Element <AssertionIDRequest>.....	41
119	3.3.2 Queries.....	41
120	3.3.2.1 Element <SubjectQuery>.....	41
121	3.3.2.2 Element <AuthnQuery>.....	42
122	3.3.2.3 Element <RequestedAuthnContext>.....	43
123	3.3.2.4 Element <AttributeQuery>.....	44
124	3.3.2.5 Element <AuthzDecisionQuery>.....	44
125	3.3.3 Element <Response>.....	45
126	3.3.4 Processing Rules.....	46
127	3.4 Authentication Request Protocol.....	46
128	3.4.1 Element <AuthnRequest>.....	47
129	3.4.1.1 Element <NameIDPolicy>.....	49
130	3.4.1.2 Element <Scoping>.....	50
131	3.4.1.3 Element <IDPList>.....	50
132	3.4.1.3.1 Element <IDPEntry>.....	51
133	3.4.1.4 Processing Rules.....	51
134	3.4.1.5 Proxying.....	52
135	3.4.1.5.1 Proxying Processing Rules.....	53
136	3.5 Artifact Resolution Protocol.....	54
137	3.5.1 Element <ArtifactResolve>.....	54
138	3.5.2 Element <ArtifactResponse>.....	55
139	3.5.3 Processing Rules.....	55
140	3.6 Name Identifier Management Protocol.....	56
141	3.6.1 Element <ManageNameIDRequest>.....	56
142	3.6.2 Element <ManageNameIDResponse>.....	57
143	3.6.3 Processing Rules.....	57
144	3.7 Single Logout Protocol.....	58
145	3.7.1 Element <LogoutRequest>.....	58
146	3.7.2 Element <LogoutResponse>.....	59
147	3.7.3 Processing Rules.....	59
148	3.7.3.1 Session Participant Rules.....	59
149	3.7.3.2 Session Authority Rules.....	60
150	3.8 Name Identifier Mapping Protocol.....	61
151	3.8.1 Element <NameIDMappingRequest>.....	61
152	3.8.2 Element <NameIDMappingResponse>.....	62
153	3.8.3 Processing Rules.....	62
154	4 SAML Versioning.....	63
155	4.1 SAML Specification Set Version.....	63
156	4.1.1 Schema Version.....	63
157	4.1.2 SAML Assertion Version.....	63
158	4.1.3 SAML Protocol Version.....	64
159	4.1.3.1 Request Version.....	64
160	4.1.4 Response Version.....	64
161	4.1.5 Permissible Version Combinations.....	65
162	4.2 SAML Namespace Version.....	65
163	4.2.1 Schema Evolution.....	65
164	5 SAML and XML Signature Syntax and Processing.....	66

165	5.1 Signing Assertions.....	66
166	5.2 Request/Response Signing.....	66
167	5.3 Signature Inheritance.....	66
168	5.4 XML Signature Profile.....	67
169	5.4.1 Signing Formats and Algorithms.....	67
170	5.4.2 References.....	67
171	5.4.3 Canonicalization Method.....	67
172	5.4.4 Transforms.....	67
173	5.4.5 KeyInfo.....	68
174	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	68
175	5.4.7 Example.....	68
176	6 SAML and XML Encryption Syntax and Processing.....	71
177	6.1 General Considerations.....	71
178	6.2 Combining Signatures and Encyption.....	71
179	7 SAML Extensibility.....	72
180	7.1 Schema Extension.....	72
181	7.1.1 Assertion Schema Extension.....	72
182	7.1.2 Protocol Schema Extension.....	72
183	7.2 Schema Wildcard Extension Points.....	73
184	7.2.1 Assertion Extension Points.....	73
185	7.2.2 Protocol Extension Points.....	73
186	7.3 Identifier Extension.....	73
187	8 SAML-Defined Identifiers.....	74
188	8.1 Action Namespace Identifiers.....	74
189	8.1.1 Read/Write/Execute/Delete/Control.....	74
190	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	74
191	8.1.3 Get/Head/Put/Post.....	75
192	8.1.4 UNIX File Permissions.....	75
193	8.2 Attribute Name Format Identifiers.....	75
194	8.2.1 Unspecified.....	75
195	8.2.2 URI Reference.....	76
196	8.2.3 Basic.....	76
197	8.3 Name Identifier Format Identifiers.....	76
198	8.3.1 Unspecified.....	76
199	8.3.2 Email Address.....	76
200	8.3.3 X.509 Subject Name.....	76
201	8.3.4 Windows Domain Qualified Name.....	76
202	8.3.5 Kerberos Principal Name.....	77
203	8.3.6 Entity Identifier.....	77
204	8.3.7 Persistent Identifier.....	77
205	8.3.8 Transient Identifier.....	78
206	8.4 Consent Identifiers.....	78
207	8.4.1 Unspecified.....	78
208	8.4.2 Obtained.....	78
209	8.4.3 Prior.....	78
210	8.4.4 Implicit.....	78
211	8.4.5 Explicit.....	79
212	8.4.6 Unavailable.....	79
213	8.4.7 Inapplicable.....	79
214	9 References.....	80
215	9.1 Normative References.....	80

216	9.2 Non-Normative References.....	80
217		

218

1 Introduction

219 The Security Assertion Markup Language (SAML) defines the syntax and processing semantics of
220 assertions made about a subject by a system entity. In the course of making, or relying upon such
221 assertions, SAML system entities may use other protocols to communicate either regarding an assertion
222 itself, or the subject of an assertion. This specification defines both the structure of SAML assertions, and
223 an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

224 SAML assertions and protocol messages are encoded in XML [XML] and use XML namespaces
225 [XMLNS]. They are typically embedded in other structures for transport, such as HTTP POST requests or
226 XML-encoded SOAP messages. The SAML bindings specification [SAMLBind] provides frameworks for
227 the embedding and transport of SAML protocol messages. The SAML profiles specification [SAMLProf]
228 provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific
229 use cases or achieve interoperability when using SAML features.

230 For general explanations of SAML terms and concepts, refer to the SAML technical overview [SAML-
231 TechOvw] and the SAML glossary [SAMLGloss]. Files containing just the SAML assertion schema [SAML-
232 XSD] and protocol schema [SAMPL-XSD] are also available.

233 The following sections describe how to understand the rest of this specification.

234 1.1 Notation

235 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
236 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
237 described in IETF RFC 2119 [RFC 2119].

238 `Listings of SAML schemas appear like this.`

239 `Example code listings appear like this.`

241 This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative
242 text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In
243 cases of disagreement between the SAML schema documents and schema listings in this specification,
244 the schema documents take precedence. Note that in some cases the normative text of this specification
245 imposes constraints beyond those indicated by the schema documents.

246 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for
247 their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is
248 present in the example:

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace, defined in a schema [SAML-XSD]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace, defined in a schema [SAMPL-XSD]. The prefix is generally elided in mentions of XML protocol-related elements in text.
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].
xenc:	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax and Processing specification [XMLEnc] and its governing schema [XMLEnc-XSD].

Prefix	XML Namespace	Comments
xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

249 This specification uses the following typographical conventions in text: <SAMLElement>,
250 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword.

251 1.2 Schema Organization and Namespaces

252 The SAML assertion structures are defined in a schema [SAML-XSD] associated with the following XML
253 namespace:

254 `urn:oasis:names:tc:SAML:2.0:assertion`

255 The SAML request-response protocol structures are defined in a schema [SAML-XP] associated with
256 the following XML namespace:

257 `urn:oasis:names:tc:SAML:2.0:protocol`

258 The assertion schema is imported into the protocol schema. See Section 4.2 for information on SAML
259 namespace versioning.

260 Also imported into both schemas is the schema for XML Signature [XMLSig], which is associated with the
261 following XML namespace:

262 `http://www.w3.org/2000/09/xmldsig#`

263 Finally, the schema for XML Encryption [XMLEnc] is imported into the assertion schema and is associated
264 with the following XML namespace:

265 `http://www.w3.org/2001/04/xmlenc#`

266 1.2.1 String Values

267 All SAML string values have the type **xs:string**, which is built in to the W3C XML Schema Datatypes
268 specification [Schema2]. Unless otherwise noted in this specification or particular profiles, all strings in
269 SAML messages MUST consist of at least one non-whitespace character (whitespace is defined in the
270 XML Recommendation [XML] §2.3).

271 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that
272 have the XML Schema **xs:string** type, or a type derived from that, MUST be compared using an exact
273 binary comparison. In particular, SAML implementations and deployments MUST NOT depend on case-
274 insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific
275 formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft
276 Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

277 If an implementation is comparing values that are represented using different character encodings, the
278 implementation MUST use a comparison method that returns the same result as converting both values to
279 the Unicode character encoding, Normalization Form C [UNICODE-C], and then performing an exact
280 binary comparison. This requirement is intended to conform to the W3C Character Model for the World
281 Wide Web [W3C-CharMod], and in particular the rules for Unicode-normalized Text.

282 Applications that compare data received in SAML documents to data from external sources MUST take
283 into account the normalization rules specified for XML. Text contained within elements is normalized so
284 that line endings are represented using linefeed characters (ASCII code 10_{Decimal}), as described in the XML
285 Recommendation [XML] §2.11. XML attribute values defined as strings (or types derived from strings) are
286 normalized as described in [XML] §3.3.3. All whitespace characters are replaced with blanks (ASCII code
287 32_{Decimal}).

288 The SAML specification does not define collation or sorting order for XML attribute values or element
289 content. SAML implementations MUST NOT depend on specific sorting orders for values, because these
290 can differ depending on the locale settings of the hosts involved.

291 1.2.2 URI Values

292 All SAML URI reference values have the type **xs:anyURI**, which is built in to the W3C XML Schema
293 Datatypes specification [Schema2].

294 Unless otherwise indicated in this specification, all URI reference values MUST consist of at least one
295 non-whitespace character, and are REQUIRED to be absolute [RFC 2396].

296 Note that the SAML specification makes extensive use of URI references as identifiers, such as status
297 codes, format types, attribute and system entity names, etc. In such cases, it is essential that the values
298 be both unique and consistent, such that the same URI is never used at different times to represent
299 different underlying information.

300 1.2.3 Time Values

301 All SAML time values have the type **xs:dateTime**, which is built in to the W3C XML Schema Datatypes
302 specification [Schema2], and MUST be expressed in UTC form, with no time zone component.

303 SAML system entities SHOULD NOT rely on time resolution finer than milliseconds. Implementations
304 MUST NOT generate time instants that specify leap seconds.

305 1.2.4 ID and ID Reference Values

306 The **xs:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses. Values
307 declared to be of type **xs:ID** in this specification MUST satisfy the following properties in addition to those
308 imposed by the definition of the **xs:ID** type itself:

- 309 • Any party that assigns an identifier MUST ensure that there is negligible probability that that party or
310 any other party will accidentally assign the same identifier to a different data object.
- 311 • Where a data object declares that it has a particular identifier, there MUST be exactly one such
312 declaration.

313 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the
314 implementation. In the case that a pseudorandom technique is employed, the probability of two randomly
315 chosen identifiers being identical MUST be less than or equal to 2^{-128} and SHOULD be less than or equal
316 to 2^{-160} . This requirement MAY be met by encoding a randomly chosen value between 128 and 160 bits in
317 length. The encoding must conform to the rules defining the **xs:ID** datatype. Such a pseudorandom
318 generator MUST be seeded with unique material in order to insure the desired uniqueness properties
319 between different systems.

320 The **xs:NCName** simple type is used in SAML to reference identifiers of type **xs:ID** since **xs>IDREF**
321 cannot be used for this purpose. In SAML, the element referred to by a SAML identifier reference might
322 actually be defined in a document separate from that in which the identifier reference is used. Using
323 **xs>IDREF** would violate the requirement that its value match the value of an ID attribute on some element
324 in the same XML document.

325 **Note:** It is anticipated that the World Wide Web Consortium will standardize a global
326 attribute for holding ID-typed values, called `xml:id` [XML-ID]. The Security Services
327 Technical Committee plans to move away from SAML-specific ID attributes to this style of
328 assigning unique identifiers as soon as practicable after the `xml:id` attribute is
329 standardized.

2 SAML Assertions

330

331 An assertion is a package of information that supplies zero or more statements made by a SAML
332 authority. SAML assertions are usually made about a **subject** (see [SAMLGloss]), represented by the
333 <Subject> element. However, the <Subject> element is optional, and other specifications and profiles
334 may utilize the SAML assertion structure to make similar statements without specifying a subject, or
335 possibly specifying the subject in an alternate way.

336 This SAML specification defines three different kinds of assertion statements that can be created by a
337 SAML authority. All SAML-defined statements are associated with a subject. The three kinds of statement
338 defined in this specification are:

- 339 • **Authentication:** The assertion subject was authenticated by a particular means at a particular time.
- 340 • **Attribute:** The assertion subject is associated with the supplied attributes.
- 341 • **Authorization Decision:** A request to allow the assertion subject to access the specified resource
342 has been granted or denied.

343 The outer structure of an assertion is generic, providing information that is common to all of the
344 statements within it. Within an assertion, a series of inner elements describe the authentication, attribute,
345 authorization decision, or user-defined statements containing the specifics.

346 As described in Section 7, extensions are permitted by the SAML assertion schema, allowing user-defined
347 extensions to assertions and statements, as well as allowing the definition of new kinds of assertions and
348 statements.

2.1 Schema Header and Namespace Declarations

349

350 The following schema fragment defines the XML namespaces and other header information for the
351 assertion schema:

```
352 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
353   xmlns="http://www.w3.org/2001/XMLSchema"  
354   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
355   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
356   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  
357   elementFormDefault="unqualified"  
358   attributeFormDefault="unqualified"  
359   blockDefault="substitution"  
360   version="2.0">  
361   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
362     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
363 schema.xsd"/>  
364   <import namespace="http://www.w3.org/2001/04/xmlenc#"  
365     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-  
366 20021210/xenc-schema.xsd"/>  
367   <annotation>  
368     <documentation>  
369       Document identifier: sstc-saml-schema-assertion-2.0  
370       Location: http://www.oasis-  
371 open.org/committees/documents.php?wg_abbrev=security  
372       Revision history:  
373       V1.0 (November, 2002):  
374         Initial Standard Schema.  
375       V1.1 (September, 2003):  
376         Updates within the same V1.0 namespace.  
377       V2.0 (August, 2004):  
378         New assertion schema based in a SAML V2.0 namespace.  
379     </documentation>  
380   </annotation>
```

381 ...
382 </schema>

383 2.2 Name Identifiers

384 The following sections define the SAML constructs that contain descriptive identifiers for subjects and for
385 assertion and message issuers.

386 There are a number of circumstances in SAML in which it is useful for two system entities need to
387 communicate regarding a third party; for example, the SAML authentication request protocol enables
388 third-party authentication of a subject. Thus, it is useful to establish a means by which parties may be
389 associated with identifiers that are meaningful to each of the parties. In some cases, it will be necessary
390 to limit the scope within which an identifier is used to a small set of system entities (to preserve the
391 privacy of a so-named subject, for example). Similar identifiers may also be used to refer to the issuer of a
392 SAML protocol message or assertion.

393 It is possible that two or more system entities may define the same name identifier to refer to different
394 identities. Thus, each entity may have a different understanding of that same name. SAML provides **name**
395 **qualifiers** to disambiguate a name identifier by effectively placing it in a federated **namespace** related to
396 the qualifiers. SAML V2.0 allows an identifier to be qualified both in terms of an asserting party and a
397 particular relying party or affiliation, allowing identifiers to exhibit pair-wise semantics, when required.

398 Name identifiers may also be encrypted to further improve their privacy-preserving characteristics,
399 particularly in cases where the identifier may be transmitted via an intermediary.

400 **Note:** To avoid use of relatively advanced XML schema constructs (among other
401 reasons), the various types of identifier elements do not share a common type hierarchy.

402 2.2.1 Element <BaseID>

403 The <BaseID> element is an extension point that allows applications to add new kinds of identifiers. Its
404 **BaseIDAbstractType** complex type is abstract and is thus usable only as the base of a derived type. It
405 includes the following attributes for use by extended identifier representations:

406 NameQualifier [Optional]

407 The security or administrative domain that qualifies the identifier. This attribute provides a means
408 to federate identifiers from disparate user stores without collision.

409 SPNameQualifier [Optional]

410 Further qualifies an identifier with the name of a service provider or affiliation of providers. This
411 attribute provides an additional means to federate identifiers on the basis of the relying party or
412 parties.

413 The following schema fragment defines the <BaseID> element and its **BaseIDType** complex type:

```
414 <attributeGroup name="IDNameQualifiers">  
415   <attribute name="NameQualifier" type="string" use="optional"/>  
416   <attribute name="SPNameQualifier" type="string" use="optional"/>  
417 </attributeGroup>  
418 <element name="BaseID" type="saml:BaseIDAbstractType"/>  
419 <complexType name="BaseIDAbstractType" abstract="true">  
420   <attributeGroup ref="saml:IDNameQualifiers"/>  
421 </complexType>
```

422 2.2.2 Element <NameID>

423 The <NameID> element is of type **NameIDType**, which permits simple string content that contains the
424 name identifier itself, and provides additional optional attributes as follows:

425 **NameQualifier** [Optional]

426 The security or administrative domain that qualifies the name. This attribute provides a means to
427 federate names from disparate user stores without collision.

428 **SPNameQualifier** [Optional]

429 Further qualifies a name with the name of a service provider or affiliation of providers. This
430 attribute provides an additional means to federate names on the basis of the relying party or
431 parties.

432 **Format** [Optional]

433 A URI reference representing the classification of string-based identifier information. See Section
434 8.3 for the SAML-defined URI references that MAY be used as the value of the **Format** attribute
435 and their associated descriptions and processing rules. If no **Format** value is provided, the value
436 `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in
437 effect.

438 When a **Format** value other than one specified in Section 8.3 is used, the content of the
439 <NameID> element is to be interpreted according to the definition of that format as provided
440 outside of this specification. If not otherwise indicated by the definition of the format, issues of
441 anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and
442 relying parties are implementation-specific.

443 **SPProvidedID** [Optional]

444 A name identifier established by a service provider or affiliation of providers for the principal, if
445 different from the primary name identifier given in the content of the <NameID> element. This
446 attribute provides a means of integrating the use of SAML with existing identifiers already in use
447 by a service provider; such an existing identifier can be "attached" to the principal using the Name
448 Identifier Management protocol defined in section 3.6.

449 The following schema fragment defines the <NameID> element and its **NameIDType** complex type:

```
450 <element name="NameID" type="saml:NameIDType"/>  
451 <complexType name="NameIDType">  
452   <simpleContent>  
453     <extension base="string"/>  
454       <attributeGroup ref="saml:IDNameQualifiers"/>  
455       <attribute name="Format" type="anyURI" use="optional"/>  
456       <attribute name="SPProvidedID" type="string" use="optional"/>  
457     </extension>  
458   </simpleContent>  
459 </complexType>
```

460 2.2.3 Element <EncryptedID>

461 The <EncryptedID> element is of type **EncryptedElementType**, and carries the content of an
462 unencrypted identifier element in encrypted fashion, as defined by the XML Encryption Syntax and
463 Processing specification [XMLEnc]. The <EncryptedID> element contains the following elements:

464 <xenc:EncryptedData> [Required]

465 The encrypted content and associated encryption details, as defined by the XML Encryption
466 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
467 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
468 encrypted content MUST contain an element that has a type that is derived from
469 **BaseIDAbstractType**, **NameIDType**, or **AssertionType**.

470 <xenc:EncryptedKey> [Zero or More]

471 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
472 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
473 the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity, as defined by
474 Section 8.3.6.

475 Encrypted identifiers are intended as a privacy protection when the plain-text value passes through an
476 intermediary. As such, the ciphertext MUST be unique to any given encryption operation. For more on
477 such issues, see [XMLEnc] §6.3.

478 The following schema fragment defines the <EncryptedID> element and its **EncryptedElementType**
479 complex type:

```
480 <complexType name="EncryptedElementType">  
481   <sequence>  
482     <element ref="xenc:EncryptedData"/>  
483     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>  
484   </sequence>  
485 </complexType>  
486 <element name="EncryptedID" type="saml:EncryptedElementType"/>
```

487 2.2.4 Element <Issuer>

488 The <Issuer> element, with complex type **NameIDType**, provides information about the issuer of a
489 SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name,
490 but permits various pieces of descriptive data. If no `Format` value is provided, the value
491 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see section 8.3.6).

492 The following schema fragment defines the <Issuer> element:

```
493 <element name="Issuer" type="saml:NameIDType"/>
```

494 2.3 Assertions

495 The following sections define the SAML constructs that either contain assertion information or provide a
496 means to refer to an existing assertion.

497 2.3.1 Element <AssertionIDRef>

498 The <AssertionIDRef> element makes a reference to a SAML assertion by its unique identifier. The
499 specific authority who issued the assertion or from whom the assertion can be obtained is not specified as
500 part of the reference.

501 The following schema fragment defines the <AssertionIDRef> element:

```
502 <element name="AssertionIDRef" type="NCName"/>
```

503 2.3.2 Element <AssertionURIRef>

504 The <AssertionURIRef> element makes a reference to a SAML assertion by URI reference. Resolving
505 the URI reference (in a fashion dictated by the URI reference itself) provides a means to retrieve the
506 assertion. See the Bindings specification [SAMLBind] for information on how this element is used in a
507 protocol binding.

508 The following schema fragment defines the <AssertionURIRef> element:

```
509 <element name="AssertionURIRef" type="anyURI"/>
```

510 2.3.3 Element <Assertion>

511 The <Assertion> element is of the **AssertionType** complex type. This type specifies the basic
512 information that is common to all assertions, including the following elements and attributes:

513 Version [Required]

514 The version of this assertion. The identifier for the version of SAML defined in this specification is
515 "2.0". SAML versioning is discussed in Section 4.

516 ID [Required]

517 The identifier for this assertion. It is of type **xs:ID**, and MUST follow the requirements specified in
518 Section 1.2.4 for identifier uniqueness.

519 IssueInstant [Required]

520 The time instant of issue in UTC, as described in Section 1.2.3.

521 <Issuer> [Required]

522 The SAML authority that is making the claim(s) in the assertion. The issuer SHOULD be unambiguous
523 to the intended relying parties.

524 This specification defines no relationship between the entity represented by this element and the
525 signer of the assertion (if any). Any such requirements imposed by a relying party that consumes the
526 assertion or by specific profiles are application-specific.

527 <ds:Signature> [Optional]

528 An XML Signature that authenticates the assertion, as described below and in section 5.

529 <Subject> [Optional]

530 The subject of the statement(s) in the assertion.

531 <Conditions> [Optional]

532 Conditions that MUST be taken into account in assessing the validity of and/or using the assertion.

533 <Advice> [Optional]

534 Additional information related to the assertion that assists processing in certain situations but which
535 MAY be ignored by applications that do not support its use.

536 Zero or more of the following statement elements:

537 <Statement>

538 A statement defined in an extension schema.

539 <AuthnStatement>

540 An authentication statement.

541 <AuthzDecisionStatement>

542 An authorization decision statement.

543 <AttributeStatement>

544 An attribute statement.

545 An assertion with no statements **MUST** contain a <Subject> element. Such an assertion identifies a
546 principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further
547 information associated with that principal.

548 Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If omitted,
549 then the statements in the assertion are assumed to identify (implicitly or explicitly) the subject or subjects
550 to which they apply in an application- or profile-specific manner. SAML itself defines no such statements,
551 and an assertion without a subject has no defined meaning in this specification.

552 Depending on the requirements of particular protocols or profiles, the issuer of a SAML assertion may
553 often need to be authenticated, and integrity protection may often be required. Authentication and
554 message integrity **MAY** be provided by mechanisms provided by a protocol binding in use during the
555 delivery of an assertion (see [SAMLBind]). The SAML assertion **MAY** be signed, which provides both
556 authentication of the issuer and integrity protection.

557 If such a signature is used, then the <ds:Signature> element **MUST** be present, and a relying party
558 **MUST** verify that the signature is valid (that is, that the assertion has not been tampered with) in
559 accordance with [XMLSig]. If it is invalid, then the relying party **MUST NOT** rely on the contents of the
560 assertion. If it is valid, then the relying party **SHOULD** evaluate the signature to determine the identity of
561 the issuer and **MAY** process the assertion in accordance with this specification and as it deems
562 appropriate.

563 The following schema fragment defines the <Assertion> element and its **AssertionType** complex type:

```
564 <element name="Assertion" type="saml:AssertionType"/>
565 <complexType name="AssertionType">
566   <sequence>
567     <element ref="saml:Issuer"/>
568     <element ref="ds:Signature" minOccurs="0"/>
569     <element ref="saml:Subject" minOccurs="0"/>
570     <element ref="saml:Conditions" minOccurs="0"/>
571     <element ref="saml:Advice" minOccurs="0"/>
572     <choice minOccurs="0" maxOccurs="unbounded">
573       <element ref="saml:Statement"/>
574       <element ref="saml:AuthnStatement"/>
575       <element ref="saml:AuthzDecisionStatement"/>
576       <element ref="saml:AttributeStatement"/>
577     </choice>
578   </sequence>
579   <attribute name="Version" type="string" use="required"/>
580   <attribute name="ID" type="ID" use="required"/>
581   <attribute name="IssueInstant" type="dateTime" use="required"/>
582 </complexType>
```

583 **2.3.4 Element <EncryptedAssertion>**

584 The <EncryptedAssertion> element represents an assertion in encrypted fashion, as defined by the
585 XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAssertion> element
586 contains the following elements:

587 <xenc:EncryptedData> [Required]
588 The encrypted content and associated encryption details, as defined by the XML Encryption
589 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
590 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
591 encrypted content MUST contain an element that has a type derived from **AssertionType**.

592 <xenc:EncryptedKey> [Zero or More]
593 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
594 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
595 the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity as defined by
596 Section 8.3.6.

597 Encrypted assertions are intended as a confidentiality protection when the plain-text value passes through
598 an intermediary.

599 The following schema fragment defines the <EncryptedAssertion> element:

```
600 <element name="EncryptedAssertion" type="saml:EncryptedElementType"/>
```

601 2.4 Subjects

602 This section defines the SAML constructs used to describe the subject of an assertion.

603 2.4.1 Element <Subject>

604 The optional <Subject> element specifies the principal that is the subject of all of the (zero or more)
605 statements in the assertion. It contains an identifier, a series of one or more subject confirmations, or
606 both:

607 <BaseID>, <NameID>, or <EncryptedID> [Optional]

608 Identifies the subject.

609 <SubjectConfirmation> [Zero or More]

610 Information that allows the subject to be confirmed. If more than one subject confirmation is provided,
611 then usage of any one of them is sufficient to confirm the subject for the purpose of applying the
612 assertion.

613 If the <Subject> element contains both an identifier and one or more subject confirmations, then the
614 SAML authority is asserting that if the SAML relying party performs the specified
615 <SubjectConfirmation>, it can treat the entity presenting the assertion to the relying party as the
616 entity that the SAML authority associates with the name identifier for the purposes of processing the
617 assertion.

618 If the <Subject> element contains only one or more subject confirmations (without an identifier), then the
619 SAML authority is asserting that if the SAML relying party performs the specified
620 <SubjectConfirmation>, it can treat the entity presenting the assertion to the relying party as the
621 entity that the SAML authority associates with the claims in the assertion for the purposes of processing
622 the assertion.

623 A <Subject> element SHOULD NOT identify more than one principal.

624 The following schema fragment defines the <Subject> element and its **SubjectType** complex type:

```
625 <element name="Subject" type="saml:SubjectType"/>  
626 <complexType name="SubjectType">  
627     <choice>
```

```

628     <sequence>
629         <choice>
630             <element ref="saml:BaseID"/>
631             <element ref="saml:NameID"/>
632             <element ref="saml:EncryptedID"/>
633         </choice>
634         <element ref="saml:SubjectConfirmation" minOccurs="0"
635 maxOccurs="unbounded"/>
636     </sequence>
637     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
638 </choice>
639 </complexType>

```

640 2.4.1.1 Element <SubjectConfirmation>

641 The <SubjectConfirmation> element provides the means for a relying party to verify the
642 correspondence of the subject of the assertion with the party with whom the relying party is
643 communicating. It contains the following attributes and elements:

644 Method [Required]

645 A URI reference that identifies a protocol to be used to confirm the subject. URI references identifying
646 SAML-defined confirmation methods are currently defined with the SAML profiles in the SAML profiles
647 specification [SAMLProf]. Additional methods MAY be added by defining new URIs and profiles or by
648 private agreement.

649 <BaseID>, <NameID>, or <EncryptedID> [Optional]

650 Identifies the entity expected to satisfy the enclosing subject confirmation requirements.

651 <SubjectConfirmationData> [Optional]

652 Additional confirmation information to be used by a specific confirmation method. For example, typical
653 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax
654 and Processing specification [XMLSig], which identifies a cryptographic key. Particular confirmation
655 methods MAY define a schema type to describe the elements, attributes, or content that may appear
656 in the <SubjectConfirmationData> element.

657 The following schema fragment defines the <SubjectConfirmation> element and its
658 **SubjectConfirmationType** complex type:

```

659 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
660 <complexType name="SubjectConfirmationType">
661     <sequence>
662         <choice minOccurs="0">
663             <element ref="saml:BaseID"/>
664             <element ref="saml:NameID"/>
665             <element ref="saml:EncryptedID"/>
666         </choice>
667         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
668     </sequence>
669     <attribute name="Method" type="anyURI" use="required"/>
670 </complexType>

```

671 2.4.1.1.1 Element <SubjectConfirmationData>

672 The <SubjectConfirmationData> element has the **SubjectConfirmationDataType** complex type. It
673 specifies additional data that allows the subject to be confirmed or constrains the circumstances under
674 which the confirmation can take place. It contains the following optional attributes that can apply to any
675 method:

676 NotBefore [Optional]

677 A time instant before which the subject cannot be confirmed. The time value is encoded in UTC, as
678 described in Section 1.2.3.

679 NotOnOrAfter [Optional]

680 A time instant at which the subject can no longer be confirmed. The time value is encoded in UTC, as
681 described in Section 1.2.3.

682 Recipient [Optional]

683 Specifies the entity or location to which an entity can present the assertion while confirming itself.

684 InResponseTo [Optional]

685 Specifies the ID of a SAML protocol message in response to which an entity can present the
686 assertion while confirming itself.

687 Address [Optional]

688 Specifies the network address from which an entity can present the assertion while authenticating
689 itself.

690 Arbitrary attributes

691 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary namespace-
692 qualified XML attributes to be added to `<SubjectConfirmationData>` constructs without the need
693 for an explicit schema extension. This allows additional fields to be added as needed to supply
694 additional confirmation-related information. SAML extensions MUST NOT add local (non-namespace-
695 qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the
696 **SubjectConfirmationDataType** complex type or a derivation of it; such attributes are reserved for
697 future maintenance and enhancement of SAML itself.

698 Arbitrary elements

699 This complex type uses an `<xs:any>` extension point to allow arbitrary XML elements to be added to
700 `<SubjectConfirmationData>` constructs without the need for an explicit schema extension. This
701 allows additional elements to be added as needed to supply additional confirmation-related
702 information.

703 Particular confirmation methods MAY require the use of one or more of the attributes defined within this
704 complex type. Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter`
705 attributes, if any, SHOULD fall within the overall assertion validity period as specified by the
706 `<Conditions>` element's `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the
707 value for `NotBefore` MUST be less than (earlier than) the value for `NotOnOrAfter`.

708 The following schema fragment defines the `<SubjectConfirmationData>` element and its
709 **SubjectConfirmationDataType** complex type:

```
710 <element name="SubjectConfirmationData"  
711 type="saml:SubjectConfirmationDataType"/>  
712 <complexType name="SubjectConfirmationDataType" mixed="true">  
713 <complexContent>  
714 <restriction base="anyType">  
715 <sequence>  
716 <any namespace="##any" processContents="lax" minOccurs="0"  
717 maxOccurs="unbounded"/>  
718 </sequence>  
719 <attribute name="NotBefore" type="dateTime" use="optional"/>  
720 <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
721 <attribute name="Recipient" type="anyURI" use="optional"/>  
722 <attribute name="InResponseTo" type="NCName" use="optional"/>  
723 <attribute name="Address" type="string" use="optional"/>  
724 <anyAttribute namespace="##other" processContents="lax"/>
```

725
726
727

```
</restriction>  
</complexContent>  
</complexType>
```

728 **2.4.1.2 Complex Type KeyInfoConfirmationDataType**

729 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`
730 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in
731 some way to authenticate the subject. The particular confirmation method **MUST** define the exact
732 mechanism by which the confirmation data can be used. The optional attributes defined by the
733 **SubjectConfirmationDataType** complex type **MAY** also appear.

734 This complex type **SHOULD** be used by any confirmation method that defines its confirmation data in
735 terms of the `<ds:KeyInfo>` element.

736 Note that in accordance with [XMLSig], each `<ds:KeyInfo>` element **MUST** identify a single
737 cryptographic key. Multiple keys **MAY** be identified with separate `<ds:KeyInfo>` elements, such as when
738 a principal uses different keys to confirm itself to different relying parties

739 The following schema fragment defines the **KeyInfoConfirmationDataType** complex type:

740
741
742
743
744
745
746
747
748

```
<complexType name="KeyInfoConfirmationDataType" mixed="false">  
  <complexContent>  
    <restriction base="saml:SubjectConfirmationDataType">  
      <sequence>  
        <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
      </sequence>  
    </restriction>  
  </complexContent>  
</complexType>
```

749 **2.5 Conditions**

750 This section defines the SAML constructs that place constraints on the acceptable use of SAML
751 assertions.

752 **2.5.1 Element <Conditions>**

753 The `<Conditions>` element **MAY** contain the following elements and attributes:

754 `NotBefore` [Optional]

755 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as
756 described in Section 1.2.3.

757 `NotOnOrAfter` [Optional]

758 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as
759 described in Section 1.2.3.

760 `<Condition>` [Any Number]

761 Provides an extension point allowing extension schemas to define new conditions.

762 `<AudienceRestriction>` [Any Number]

763 Specifies that the assertion is addressed to a particular audience.

764 <OneTimeUse> [Optional]

765 Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future
766 use. Although the schema permits multiple occurrences, there MUST be at most one instance of
767 this element.

768 <ProxyRestriction> [Optional]

769 Specifies limitations that the asserting party imposes on relying parties that wish to subsequently act
770 as asserting parties themselves and issue assertions of their own on the basis of the information
771 contained in the original assertion. Although the schema permits multiple occurrences, there MUST
772 be at most one instance of this element.

773 Because the use of the `xsi:type` attribute would permit an assertion to contain more than one instance
774 of a SAML-defined subtype of **ConditionsType** (such as **OneTimeUseType**), the schema does not
775 explicitly limit the number of times particular conditions may be included. A particular type of condition
776 MAY define limits on such use, as shown above.

777 The following schema fragment defines the <Conditions> element and its **ConditionsType** complex
778 type:

```
779 <element name="Conditions" type="saml:ConditionsType"/>  
780 <complexType name="ConditionsType">  
781   <choice minOccurs="0" maxOccurs="unbounded">  
782     <element ref="saml:Condition"/>  
783     <element ref="saml:AudienceRestriction"/>  
784     <element ref="saml:OneTimeUse"/>  
785     <element ref="saml:ProxyRestriction"/>  
786   </choice>  
787   <attribute name="NotBefore" type="dateTime" use="optional"/>  
788   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
789 </complexType>
```

790 If an assertion contains a <Conditions> element, the validity of the assertion is dependent on the sub-
791 elements and attributes provided, using the following rules in the order shown below.

792 Note that an assertion that has condition validity status **Valid** may nonetheless be untrustworthy or invalid
793 for reasons such as not being well-formed or schema-valid, not being issued by a trustworthy SAML
794 authority, or not being authenticated by a trustworthy means.

795 Also note that some conditions may not directly impact the validity of the containing assertion (they always
796 evaluate to **Valid**), but may restrict the behavior of relying parties with respect to the use of the assertion.

- 797 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is
798 considered to be **Valid** with respect to condition processing.
- 799 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the
800 assertion is considered to be **Invalid**.
- 801 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity
802 of the assertion cannot be determined and is considered to be **Indeterminate**.
- 803 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the
804 assertion is considered to be **Valid** with respect to condition processing.

805 The <Conditions> element MAY be extended to contain additional conditions. If an element contained
806 within a <Conditions> element is encountered that is not understood, the status of the condition cannot
807 be evaluated and the validity status of the assertion MUST be considered to be **Indeterminate** in
808 accordance with rule 3 above.

809 **2.5.1.1 Attributes NotBefore and NotOnOrAfter**

810 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion within
811 the context of its profile(s) of use. They do not guarantee that the statements in the assertion will be valid
812 throughout the validity period.

813 The `NotBefore` attribute specifies the time instant at which the validity interval begins. The
814 `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.

815 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
816 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the
817 assertion is valid with respect to conditions at any time before the time instant specified by the
818 `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that
819 are supplied evaluate to **Valid**), the assertion is valid with respect to conditions from the time instant
820 specified by the `NotBefore` attribute with no expiry. If neither attribute is specified (and if any other
821 conditions that are supplied evaluate to **Valid**), the assertion is valid with respect to conditions at any time.

822 If both attributes are present, the value for `NotBefore` MUST be less than (earlier than) the value for
823 `NotOnOrAfter`.

824 **2.5.1.2 Element <Condition>**

825 The `<Condition>` element serves as an extension point for new conditions. Its `ConditionAbstractType`
826 complex type is abstract and is thus usable only as the base of a derived type.

827 The following schema fragment defines the `<Condition>` element and its `ConditionAbstractType`
828 complex type:

```
829 <element name="Condition" type="saml:ConditionAbstractType"/>  
830 <complexType name="ConditionAbstractType" abstract="true"/>
```

831 **2.5.1.3 Elements <AudienceRestriction> and <Audience>**

832 The `<AudienceRestriction>` element specifies that the assertion is addressed to one or more
833 specific audiences identified by `<Audience>` elements. Although a SAML relying party that is outside the
834 audiences specified is capable of drawing conclusions from an assertion, the SAML authority explicitly
835 makes no representation as to accuracy or trustworthiness to such a party. It contains the following
836 element:

837 `<Audience>`

838 A URI reference that identifies an intended audience. The URI reference MAY identify a document
839 that describes the terms and conditions of audience membership. It MAY also contain the unique
840 identifier of a SAML system entity, as described by the name identifier `Format` URI of
841 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

842 The audience restriction condition evaluates to **Valid** if and only if the SAML relying party is a member of
843 one or more of the audiences specified.

844 The SAML authority cannot prevent a party to whom the assertion is disclosed from taking action on the
845 basis of the information provided. However, the `<AudienceRestriction>` element allows the SAML
846 authority to state explicitly that no warranty is provided to such a party in a machine- and human-readable
847 form. While there can be no guarantee that a court would uphold such a warranty exclusion in every
848 circumstance, the probability of upholding the warranty exclusion is considerably improved.

849 Note that multiple `<AudienceRestriction>` elements MAY be included in a single assertion, and each
850 MUST be evaluated independently.

851 The following schema fragment defines the <AudienceRestriction> element and its
852 **AudienceRestrictionType** complex type:

```
853 <element name="AudienceRestriction"  
854 type="saml:AudienceRestrictionType"/>  
855 <complexType name="AudienceRestrictionType">  
856 <complexContent>  
857 <extension base="saml:ConditionAbstractType">  
858 <sequence>  
859 <element ref="saml:Audience" maxOccurs="unbounded"/>  
860 </sequence>  
861 </extension>  
862 </complexContent>  
863 </complexType>  
864 <element name="Audience" type="anyURI"/>
```

865 **2.5.1.4 Element <OneTimeUse>**

866 In general, relying parties may choose to retain assertions, or the information they contain in some other
867 form, for reuse. The <OneTimeUse> condition element allows an authority to indicate that the information
868 in the assertion is likely to change very soon and fresh information should be obtained for each use. An
869 example would be an assertion containing an <AuthzDecisionStatement> which was the result of a
870 policy which specified access control which was a function of the time of day.

871 If system clocks in a distributed environment could be precisely synchronized, then this requirement could
872 be met by careful use of the validity interval. However, since some clock skew between systems will
873 always be present and will be combined with possible transmission delays, there is no convenient way for
874 the issuer to appropriately limit the lifetime of an assertion without running a substantial risk that it will
875 already have expired before it arrives.

876 The <OneTimeUse> element indicates that the assertion SHOULD be used immediately by the relying
877 party and MUST NOT be retained for future use. Relying parties are always free to request a fresh
878 assertion for every use. However, implementations that choose to retain assertions for future use MUST
879 observe the <OneTimeUse> element. This condition is independent from the *NotBefore* and
880 *NotOnOrAfter* condition information.

881 To support the single use constraint, a relying party should maintain a cache of the assertions it has
882 processed containing such a condition. Whenever an assertion with this condition is processed, the cache
883 should be checked to ensure that the same assertion has not been previously received and processed by
884 the relying party.

885 A SAML authority MUST NOT include more than one <OneTimeUse> element within a <Conditions>
886 element of an assertion.

887 For the purposes of determining the validity of the <Conditions> element, the <OneTimeUse> is
888 considered to always be valid. (That is, this condition does not affect validity but is a condition on use.)

889 The following schema fragment defines the <OneTimeUse> element and its **OneTimeUseType** complex
890 type:

```
891 <element name="OneTimeUse" type="saml:OneTimeUseType"/>  
892 <complexType name="OneTimeUseType">  
893 <complexContent>  
894 <extension base="saml:ConditionAbstractType"/>  
895 </complexContent>  
896 </complexType>
```

897 **2.5.1.5 Element <ProxyRestriction>**

898 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent
899 assertions of their own on the basis of the information contained in the original assertion. A relying party
900 MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis of
901 an assertion containing such a condition.

902 The <ProxyRestriction> element contains the following elements and attributes:

903 Count [Optional]

904 Specifies the number of indirections that MAY exist between this assertion and an assertion which has
905 ultimately been issued on the basis of it.

906 <Audience> [Zero or More]

907 Specifies the set of audiences to whom new assertions MAY be issued on the basis of this assertion.

908 A Count value of zero indicates that a relying party MUST NOT issue an assertion to another relying party
909 on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves contain a
910 <ProxyRestriction> element with a Count value of at most one less than this value.

911 If no <Audience> elements are specified, then no audience restrictions are imposed on the relying
912 parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST
913 themselves contain an <AudienceRestriction> element with at least one of the <Audience>
914 elements present in the previous <ProxyRestriction> element, and no <Audience> elements
915 present that were not in the previous <ProxyRestriction> element.

916 A SAML authority MUST NOT include more than one <ProxyRestriction> element within a
917 <Conditions> element of an assertion.

918 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>
919 condition is considered to always be valid. (That is, this condition does not affect validity but is a condition
920 on use.)

921 The following schema fragment defines the <ProxyRestriction> element and its

922 **ProxyRestrictionType** complex type:

```
923 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
924 <complexType name="ProxyRestrictionType">
925   <complexContent>
926     <extension base="saml:ConditionAbstractType">
927       <sequence>
928         <element ref="saml:Audience" minOccurs="0"
929 maxOccurs="unbounded"/>
930       </sequence>
931       <attribute name="Count" type="nonNegativeInteger" use="optional"/>
932     </extension>
933   </complexContent>
934 </complexType>
```

935 **2.6 Advice**

936 This section defines the SAML constructs that contain additional information about an assertion that a
937 SAML authority wishes to provide to an assertion consumer.

938 2.6.1 Element <Advice>

939 The <Advice> element contains any additional information that the SAML authority wishes to provide.
940 This information MAY be ignored by applications without affecting either the semantics or the validity of
941 the assertion.

942 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,
943 <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in
944 other non-SAML namespaces. If elements from non-SAML namespaces are used, lax schema validation
945 must be used when processing the elements.

946 Following are some potential uses of the <Advice> element:

- 947 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating
948 the claims) or indirectly (by reference to the supporting assertions).
- 949 • State a proof of the assertion claims.
- 950 • Specify the timing and distribution points for updates to the assertion.

951 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
952 <element name="Advice" type="saml:AdviceType"/>  
953 <complexType name="AdviceType">  
954   <choice minOccurs="0" maxOccurs="unbounded">  
955     <element ref="saml:AssertionIDRef"/>  
956     <element ref="saml:AssertionURIRef"/>  
957     <element ref="saml:Assertion"/>  
958     <element ref="saml:EncryptedAssertion"/>  
959     <any namespace="##other" processContents="lax"/>  
960   </choice>  
961 </complexType>
```

962 2.7 Statements

963 The following sections define the SAML constructs that contain statement information.

964 2.7.1 Element <Statement>

965 The <Statement> element is an extension point that allows other assertion-based applications to reuse
966 the SAML assertion framework. SAML itself derives its core statements from this extension point. Its
967 **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

968 The following schema fragment defines the <Statement> element and its **StatementAbstractType**
969 complex type:

```
970 <element name="Statement" type="saml:StatementAbstractType"/>  
971 <complexType name="StatementAbstractType" abstract="true"/>
```

972 2.7.2 Element <AuthnStatement>

973 The <AuthnStatement> element describes a statement by the SAML authority asserting that the
974 assertion subject was authenticated by a particular means at a particular time. Assertions containing
975 <AuthnStatement> elements MUST contain a <Subject> element.

976 It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the
977 following elements and attributes:

978 **Note:** The <AuthorityBinding> element and its corresponding type were removed
979 from <AuthnStatement> for V2.0 of SAML.

980 AuthnInstant [Required]

981 Specifies the time at which the authentication took place. The time value is encoded in UTC, as
982 described in Section 1.2.3.

983 SessionIndex [Optional]

984 Indexes a particular session between the subject and the authenticating authority.

985 SessionNotOnOrAfter [Optional]

986 Specifies a time instant at which the session between the subject and the authority issuing this
987 statement MUST be considered ended. The time value is encoded in UTC, as described in Section
988 1.2.3. There is no required relationship between this attribute and a NotOnOrAfter condition
989 attribute that may be present in the assertion.

990 <SubjectLocality> [Optional]

991 Specifies the DNS domain name and IP address for the system from which the assertion subject was
992 apparently authenticated.

993 <AuthnContext> [Required]

994 The context used by the identity provider up to and including the authentication event that yielded this
995 statement. Contains a reference to an authentication context class, an authentication context
996 declaration, declaration reference, or both. See the Authentication Context specification
997 [SAMLAuthnCxt] for a full description of authentication context information.

998 In general, any string value MAY be used as a SessionIndex value. However, when privacy is a
999 consideration, care must be taken to ensure that the SessionIndex value does not invalidate other
1000 privacy mechanisms. In such cases, the value MUST NOT be usable to correlate activity by a principal
1001 across different session participants. Two solutions that achieve this goal are provided below and are
1002 RECOMMENDED:

- 1003 • Use small positive integers (or reoccurring constants in a list) for the SessionIndex. The authority
1004 SHOULD choose the range of values such that the cardinality of any one integer will be sufficiently high
1005 to prevent a particular principal's actions from being correlated across multiple session participants.
1006 The authority SHOULD choose values for SessionIndex randomly from within this range (except
1007 when required to ensure unique values for subsequent statements given to the same session
1008 participant but as part of a distinct session).
- 1009 • Use the enclosing assertion's ID value in the SessionIndex.

1010 The following schema fragment defines the <AuthnStatement> element and its AuthnStatementType
1011 complex type:

```
1012 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
1013 <complexType name="AuthnStatementType">
1014   <complexContent>
1015     <extension base="saml:StatementAbstractType">
1016       <sequence>
1017         <element ref="saml:SubjectLocality" minOccurs="0"/>
1018         <element ref="saml:AuthnContext"/>
1019       </sequence>
1020       <attribute name="AuthnInstant" type="dateTime" use="required"/>
1021       <attribute name="SessionIndex" type="string" use="optional"/>
1022       <attribute name="SessionNotOnOrAfter" type="dateTime"
1023 use="optional"/>
1024     </extension>
1025   </complexContent>
```

1026 </complexType>

1027 2.7.2.1 Element <SubjectLocality>

1028 The <SubjectLocality> element specifies the DNS domain name and IP address for the system from
1029 which the assertion subject was authenticated. It has the following attributes:

1030 Address [Optional]

1031 The network address of the system from which the subject was authenticated.

1032 DNSName [Optional]

1033 The DNS name of the system from which the subject was authenticated.

1034 This element is entirely advisory, since both of these fields are quite easily “spoofed,” but may be useful
1035 information in some applications.

1036 The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**
1037 complex type:

```
1038 <element name="SubjectLocality"  
1039     type="saml: SubjectLocalityType"/>  
1040 <complexType name="SubjectLocalityType">  
1041     <attribute name="Address" type="string" use="optional"/>  
1042     <attribute name="DNSName" type="string" use="optional"/>  
1043 </complexType>
```

1044 2.7.2.2 Element <AuthnContext>

1045 The <AuthnContext> element specifies the context of an authentication event with an authentication
1046 context class reference, an authentication context declaration or declaration reference, or both. Its
1047 complex **AuthnContextType** has the following elements:

1048 <AuthnContextClassRef> [Optional]

1049 A URI reference identifying an authentication context class that describes the authentication context
1050 declaration that follows.

1051 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

1052 Either an authentication context declaration provided by value, or a URI reference that identifies such
1053 a declaration. The URI reference MAY directly resolve into an XML document containing the
1054 referenced declaration.

1055 <AuthenticatingAuthority> [Zero or More]

1056 Zero or more unique identifiers of authentication authorities that were involved in the authentication of
1057 the principal in addition to the assertion issuer.

1058 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**
1059 complex type:

```
1060 <element name="AuthnContext" type="saml:AuthnContextType"/>  
1061 <complexType name="AuthnContextType">  
1062     <sequence>  
1063         <choice>  
1064             <sequence>  
1065                 <element ref="saml:AuthnContextClassRef"/>  
1066                 <choice minOccurs="0">  
1067                     <element ref="saml:AuthnContextDecl"/>  
1068                     <element ref="saml:AuthnContextDeclRef"/>  
1069                 </choice>  
1069     </sequence>
```

```

1070         </sequence>
1071         <choice>
1072             <element ref="saml:AuthnContextDecl"/>
1073             <element ref="saml:AuthnContextDeclRef"/>
1074         </choice>
1075     </choice>
1076     <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1077     maxOccurs="unbounded"/>
1078 </sequence>
1079 </complexType>
1080 <element name="AuthnContextClassRef" type="anyURI"/>
1081 <element name="AuthnContextDeclRef" type="anyURI"/>
1082 <element name="AuthnContextDecl" type="anyType"/>
1083 <element name="AuthenticatingAuthority" type="anyURI"/>

```

1084 2.7.3 Element <AttributeStatement>

1085 The <AttributeStatement> element describes a statement by the SAML authority asserting that the
1086 assertion subject is associated with the specified attributes. Assertions containing
1087 <AttributeStatement> elements MUST contain a <Subject> element.

1088 It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the
1089 following elements:

1090 <Attribute> or <EncryptedAttribute> [One or More]

1091 The <Attribute> element specifies an attribute of the subject. An encrypted SAML attribute may be
1092 included with the <EncryptedAttribute> element.

1093 The following schema fragment defines the <AttributeStatement> element and its
1094 **AttributeStatementType** complex type:

```

1095 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1096 <complexType name="AttributeStatementType">
1097     <complexContent>
1098         <extension base="saml:StatementAbstractType">
1099             <choice maxOccurs="unbounded">
1100                 <element ref="saml:Attribute"/>
1101                 <element ref="saml:EncryptedAttribute"/>
1102             </choice>
1103         </extension>
1104     </complexContent>
1105 </complexType>

```

1106 2.7.3.1 Element <Attribute>

1107 The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the
1108 **AttributeType** complex type. It is used within an attribute statement to express particular attributes and
1109 values associated with an assertion subject, as described in the previous section. It is also used in an
1110 attribute query to request that the values of specific SAML attributes be returned (see section 3.3.2.4 for
1111 more information). The <Attribute> element contains the following XML attributes:

1112 Name [Required]

1113 The name of the attribute.

1114 NameFormat [Optional]

1115 A URI reference representing the classification of the attribute name for purposes of interpreting the
1116 name. See section 8.2 for some URI references that MAY be used as the value of the NameFormat
1117 attribute and their associated descriptions and processing rules. If no NameFormat value is provided,
1118 the identifier urn:oasis:names:tc:SAML:2.0:attname-format:unspecified (see section

1119 8.2.1) is in effect.

1120 FriendlyName [Optional]

1121 A string that provides a more human-readable form of the attribute's name, which may be useful in
1122 cases in which the actual Name is complex or opaque, such as an OID or a UUID. This value MUST
1123 NOT be used as a basis for formally identifying SAML attributes.

1124 Arbitrary attributes

1125 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to
1126 be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows
1127 additional fields to be added as needed to supply additional parameters to be used, for example, in an
1128 attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or
1129 XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a
1130 derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1131 `<AttributeValue>` [Any Number]

1132 Contains a value of the attribute. If an attribute contains more than one discrete value, it is
1133 RECOMMENDED that each value appear in its own `<AttributeValue>` element. If more than
1134 one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a
1135 datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have
1136 the identical datatype assigned.

1137 The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on
1138 its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the
1139 `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of
1140 values indicates that the requester is interested in any or all of the named attribute's values (see also
1141 section 3.3.2.4).

1142 Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the
1143 semantics of specifying or omitting `<AttributeValue>` elements.

1144 The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
1145 <element name="Attribute" type="saml:AttributeType"/>
1146 <complexType name="AttributeType">
1147   <sequence>
1148     <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
1149   </sequence>
1150   <attribute name="Name" type="string" use="required"/>
1151   <attribute name="NameFormat" type="anyURI" use="optional"/>
1152   <attribute name="FriendlyName" type="string" use="optional"/>
1153   <anyAttribute namespace="##other" processContents="lax"/>
1154 </complexType>
```

1155 2.7.3.1.1 Element `<AttributeValue>`

1156 The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the
1157 **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

1158 If the data content of an `<AttributeValue>` element is of an XML Schema simple type (such as
1159 **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration
1160 in the `<AttributeValue>` element. If the attribute value contains structured data, the necessary data
1161 elements MAY be defined in an extension schema.

1162 **Note:** Specifying a datatype other than an XML Schema simple type on
1163 `<AttributeValue>` using `xsi:type` will require the presence of the extension schema
1164 that defines the datatype in order for schema processing to proceed.

1165 If a SAML attribute includes an empty value, such as the empty string, the corresponding
1166 <AttributeValue> element MUST be empty (generally this is serialized as <AttributeValue/>).
1167 This overrides the requirement in section 1.2.1 that string values in SAML content contain at least one
1168 non-whitespace character.

1169 If a SAML attribute includes a "null" value, the corresponding <AttributeValue> element MUST be
1170 empty and MUST contain the reserved xsi:nil XML attribute with a value of "true" or "1".

1171 The following schema fragment defines the <AttributeValue> element:

```
1172 <element name="AttributeValue" type="anyType" nillable="true"/>
```

1173 2.7.3.2 Element <EncryptedAttribute>

1174 The <EncryptedAttribute> element represents a SAML attribute in encrypted fashion, as defined by
1175 the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAttribute>
1176 element contains the following elements:

1177 <xenc:EncryptedData> [Required]

1178 The encrypted content and associated encryption details, as defined by the XML Encryption
1179 Syntax and Processing specification [XMLEnc]. The Type attribute SHOULD be present and, if
1180 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
1181 encrypted content MUST contain an element that has a type that is derived from **AttributeType**.

1182 <xenc:EncryptedKey> [Zero or More]

1183 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
1184 Recipient attribute that specifies the entity for whom the key has been encrypted. The value of
1185 the Recipient attribute SHOULD be the URI identifier of a SAML system entity as defined by
1186 Section 8.3.6.

1187 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through
1188 an intermediary.

1189 The following schema fragment defines the <EncryptedAttribute> element:

```
1190 <element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

1191 2.7.4 Element <AuthzDecisionStatement>

1192 **Note:** The <AuthzDecisionStatement> feature has been frozen as of SAML V2.0,
1193 with no future enhancements planned. Users who require additional functionality may
1194 want to consider the eXtensible Access Control Markup Language [XACML], which offers
1195 enhanced authorization decision features.

1196 The <AuthzDecisionStatement> element describes a statement by the SAML authority asserting that
1197 a request for access by the assertion subject to the specified resource has resulted in the specified
1198 authorization decision on the basis of some optionally specified evidence. Assertions containing
1199 <AuthzDecisionStatement> elements MUST contain a <Subject> element.

1200 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1201 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
1202 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
1203 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1204 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1205 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1206 dependent. When a scheme uses elements of the common syntax, it will also use the common

1207 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1208 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where
1209 the port is elided.

1210 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the
1211 URI normalized form wherever possible as follows:

- 1212 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1213 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1214 Inconsistent URI reference interpretation can also result from differences between the URI reference
1215 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1216 employed to specify an access control policy language. The following security conditions SHOULD be
1217 satisfied by the system which employs SAML assertions:

- 1218 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
1219 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
1220 part of the resource URI reference.
- 1221 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
1222 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
1223 gain access to a denied resource by creating such an equivalence.

1224 The `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends
1225 **StatementAbstractType** with the addition of the following elements and attributes:

1226 **Resource** [Required]

1227 A URI reference identifying the resource to which access authorization is sought. This attribute MAY
1228 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the
1229 current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1230 **Decision** [Required]

1231 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1232 the **DecisionType** simple type.

1233 **<Action>** [One or more]

1234 The set of actions authorized to be performed on the specified resource.

1235 **<Evidence>** [Optional]

1236 A set of assertions that the SAML authority relied on in making the decision.

1237 The following schema fragment defines the `<AuthzDecisionStatement>` element and its
1238 **AuthzDecisionStatementType** complex type:

```
1239 <element name="AuthzDecisionStatement"  
1240 type="saml:AuthzDecisionStatementType"/>  
1241 <complexType name="AuthzDecisionStatementType">  
1242   <complexContent>  
1243     <extension base="saml:StatementAbstractType">  
1244       <sequence>  
1245         <element ref="saml:Action" maxOccurs="unbounded"/>  
1246         <element ref="saml:Evidence" minOccurs="0"/>  
1247       </sequence>  
1248       <attribute name="Resource" type="anyURI" use="required"/>  
1249       <attribute name="Decision" type="saml:DecisionType" use="required"/>  
1250     </extension>  
1251   </complexContent>  
1252 </complexType>
```

1253 **2.7.4.1 Simple Type DecisionType**

1254 The **DecisionType** simple type defines the possible values to be reported as the status of an
1255 authorization decision statement.

1256 Permit

1257 The specified action is permitted.

1258 Deny

1259 The specified action is denied.

1260 Indeterminate

1261 The SAML authority cannot determine whether the specified action is permitted or denied.

1262 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to
1263 provide an affirmative statement that it is not able to issue a decision. Additional information as to the
1264 reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>` elements in
1265 the enclosing `<Response>`.

1266 The following schema fragment defines the **DecisionType** simple type:

```
1267 <simpleType name="DecisionType">  
1268   <restriction base="string">  
1269     <enumeration value="Permit"/>  
1270     <enumeration value="Deny"/>  
1271     <enumeration value="Indeterminate"/>  
1272   </restriction>  
1273 </simpleType>
```

1274 **2.7.4.2 Element <Action>**

1275 The `<Action>` element specifies an action on the specified resource for which permission is sought. Its
1276 string-data content provides the label for an action sought to be performed on the specified resource, and
1277 it has the following attribute:

1278 Namespace [Optional]

1279 A URI reference representing the namespace in which the name of the specified action is to be
1280 interpreted. If this element is absent, the namespace

1281 `urn:oasis:names:tc:SAML:1.0:action:rwdc-negation` specified in Section 8.1.2 is in
1282 effect.

1283 The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```
1284 <element name="Action" type="saml:ActionType"/>  
1285 <complexType name="ActionType">  
1286   <simpleContent>  
1287     <extension base="string">  
1288       <attribute name="Namespace" type="anyURI" use="required"/>  
1289     </extension>  
1290   </simpleContent>  
1291 </complexType>
```

1292 **2.7.4.3 Element <Evidence>**

1293 The `<Evidence>` element contains an assertion or assertion reference that the SAML authority relied on
1294 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one
1295 or more of the following elements:

- 1296 <AssertionIDRef> [Any number]
1297 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.
- 1298 <AssertionURIRef> [Any number]
1299 Specifies an assertion by means of a URI reference.
- 1300 <Assertion> [Any number]
1301 Specifies an assertion by value.
- 1302 <EncryptedAssertion> [Any number]
1303 Specifies an encrypted assertion by value.
- 1304 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1305 and the SAML authority making the authorization decision. For example, in the case that the SAML relying
1306 party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1307 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1308 assertion as valid either to the relying party or any other third party.

1309 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
1310 <element name="Evidence" type="saml:EvidenceType"/>  
1311 <complexType name="EvidenceType">  
1312   <choice maxOccurs="unbounded">  
1313     <element ref="saml:AssertionIDRef"/>  
1314     <element ref="saml:AssertionURIRef"/>  
1315     <element ref="saml:Assertion"/>  
1316     <element ref="saml:EncryptedAssertion"/>  
1317   </choice>  
1318 </complexType>
```

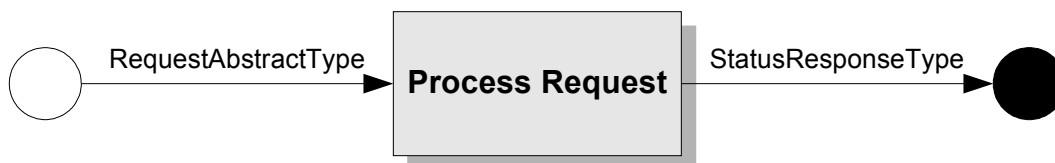
3 SAML Protocols

1319

1320 SAML protocol messages MAY be generated and exchanged using a variety of protocols. The SAML
1321 bindings specification [SAMLBind] describes specific means of transporting protocol messages using
1322 existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a
1323 number of applications of the protocols defined in this section together with additional processing rules,
1324 restrictions, and requirements that facilitate interoperability.

1325 Specific SAML request and response messages derive from common types. The requester sends an
1326 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1327 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1328



1330

Figure 1: SAML Request-Response Protocol

1331 The protocols defined by SAML achieve the following actions:

- 1332 • Returning one or more requested assertions (includes a direct request of the desired assertions, as
1333 well as querying for assertions that meet particular criteria)
- 1334 • Performing authentication on request and returning the corresponding assertion
- 1335 • Registering a name identifier or terminating a name registration on request
- 1336 • Retrieving a protocol message that has been requested by means of an artifact
- 1337 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
1338 request
- 1339 • Providing a name identifier mapping on request

1340 Throughout this section, text mentions of elements and types in the SAML protocol namespace are not
1341 shown with the conventional namespace prefix `samlp:`. For clarity, text mentions of elements and types
1342 in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:`.

3.1 Schema Header and Namespace Declarations

1343

1344 The following schema fragment defines the XML namespaces and other header information for the
1345 protocol schema:

```
1346 <schema  
1347   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1348   xmlns="http://www.w3.org/2001/XMLSchema"  
1349   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1350   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1351   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"   
1352   elementFormDefault="unqualified"  
1353   attributeFormDefault="unqualified"  
1354   blockDefault="substitution"  
1355   version="2.0">  
1356   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
1357     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
```

```

1358     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1359           schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
1360 schema.xsd"/>
1361     <annotation>
1362       <documentation>
1363         Document identifier: draft-sstc-saml-schema-protocol-2.0-01
1364         Location: http://www.oasis-
1365 open.org/committees/documents.php?wg_abbrev=security
1366         Revision history:
1367         V1.0 (November, 2002):
1368           Initial Standard Schema.
1369         V1.1 (September, 2003):
1370           Updates within the same V1.0 namespace.
1371         V2.0 (August, 2004):
1372           New protocol schema based in a SAML V2.0 namespace.
1373       </documentation>
1374     </annotation>
1375     ...
1376 </schema>

```

1377 3.2 Requests and Responses

1378 The following sections define the SAML constructs and basic requirements that underlie all of the request
1379 and response messages used in SAML protocols.

1380 3.2.1 Complex Type RequestAbstractType

1381 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1382 This type defines common attributes and elements that are associated with all SAML requests:

1383 **Note:** The `<RespondWith>` element has been removed from **RequestAbstractType**
1384 for V2.0 of SAML.

1385 ID [Required]

1386 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section
1387 1.2.4 for identifier uniqueness. The values of the `ID` attribute in a request and the `InResponseTo`
1388 attribute in the corresponding response MUST match.

1389 Version [Required]

1390 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".
1391 SAML versioning is discussed in Section 4.

1392 IssueInstant [Required]

1393 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
1394 1.2.3.

1395 Destination [Optional]

1396 A URI reference indicating the address to which this request has been sent. This is useful to prevent
1397 malicious forwarding of requests to unintended recipients, a protection that is required by some
1398 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1399 location at which the message was received. If it does not, the request MUST be discarded. Some
1400 protocol bindings may require the use of this attribute (see [SAMLBind]).

1401 Consent [Optional]

1402 Indicates whether or not (and under what conditions) consent has been obtained from a user in the
1403 sending this request. See Section 8.4 for some URI references that MAY be used as the value of the
1404 `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the identifier

1405 urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in effect.

1406 <saml:Issuer> [Optional]

1407 Identifies the entity that generated the request message.

1408 <ds:Signature> [Optional]

1409 An XML Signature that authenticates the requester and provides message integrity, as described
1410 below and in Section 5.

1411 <Extensions> [Optional]

1412 This extension point contains optional protocol message extension elements that are agreed on
1413 between the communicating parties. No extension schema is required in order to make use of this
1414 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1415 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1416 SAML-defined namespace.

1417 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to
1418 authenticate itself, and message integrity may often be required. Authentication and message integrity
1419 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request
1420 MAY be signed, which provides both authentication of the requester and message integrity.

1421 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1422 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)
1423 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the
1424 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the
1425 signature to determine the identity of the signer and MAY process the request or respond with an error (if
1426 the request is invalid for some other reason).

1427 If a Consent attribute is included and the value indicates that some form of user consent has been
1428 obtained, then the request SHOULD be signed.

1429 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if
1430 it responds, it MUST return a SAML response message with a <StatusCode> element with the value
1431 urn:oasis:names:tc:SAML:2.0:status:Requester.

1432 The following schema fragment defines the **RequestAbstractType** complex type:

```

1433 <complexType name="RequestAbstractType" abstract="true">
1434   <sequence>
1435     <element ref="saml:Issuer" minOccurs="0"/>
1436     <element ref="ds:Signature" minOccurs="0"/>
1437     <element ref="samlp:Extensions" minOccurs="0"/>
1438   </sequence>
1439   <attribute name="ID" type="ID" use="required"/>
1440   <attribute name="Version" type="string" use="required"/>
1441   <attribute name="IssueInstant" type="dateTime" use="required"/>
1442   <attribute name="Destination" type="anyURI" use="optional"/>
1443   <attribute name="Consent" type="anyURI" use="optional"/>
1444 </complexType>
1445 <element name="Extensions" type="samlp:ExtensionsType"/>
1446 <complexType name="ExtensionsType">
1447   <sequence>
1448     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1449   </sequence>
1450 </complexType>

```

1451 3.2.2 Complex Type StatusResponseType

1452 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
1453 defines common attributes and elements that are associated with all SAML responses:

1454 ID [Required]
1455 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in
1456 Section 1.2.4 for identifier uniqueness.

1457 InResponseTo [Optional]
1458 A reference to the identifier of the request to which the response corresponds, if any. If the response
1459 is not generated in response to a request, or if the ID attribute value of a request cannot be
1460 determined (because the request is malformed), then this attribute MUST NOT be present. Otherwise,
1461 it MUST be present and its value MUST match the value of the corresponding request's ID attribute
1462 value.

1463 Version [Required]
1464 The version of this response. The identifier for the version of SAML defined in this specification is
1465 "2.0". SAML versioning is discussed in Section 4.

1466 IssueInstant [Required]
1467 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
1468 1.2.3.

1469 Destination [Optional]
1470 A URI reference indicating the address to which this response has been sent. This is useful to prevent
1471 malicious forwarding of responses to unintended recipients, a protection that is required by some
1472 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1473 location at which the message was received. If it does not, the response MUST be discarded. Some
1474 protocol bindings may require the use of this attribute (see [SAMLBind]).

1475 <saml:Issuer> [Optional]
1476 Identifies the entity that generated the response message.

1477 <ds:Signature> [Optional]
1478 An XML Signature that authenticates the responder and provides message integrity, as described
1479 below and in Section 5.

1480 <Extensions> [Optional]
1481 This contains optional protocol message extension elements that are agreed on between the
1482 communicating parties. . No extension schema is required in order to make use of this extension
1483 point, and even if one is provided, the lax validation setting does not impose a requirement for the
1484 extension to be valid. SAML extension elements MUST be namespace-qualified in a non-SAML-
1485 defined namespace.

1486 <Status> [Required]
1487 A code representing the status of the corresponding request.

1488 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to
1489 authenticate itself, and message integrity may often be required. Authentication and message integrity
1490 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML
1491 response MAY be signed, which provides both authentication of the responder and message integrity.

1492 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1493 requester receiving the response MUST verify that the signature is valid (that is, that the message has not
1494 been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on
1495 the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD
1496 evaluate the signature to determine the identity of the signer and MAY process the response as it deems
1497 appropriate.

1498 The following schema fragment defines the **StatusResponseType** complex type:

```

1499 <complexType name="StatusResponseType">
1500   <sequence>
1501     <element ref="saml:Issuer" minOccurs="0"/>
1502     <element ref="ds:Signature" minOccurs="0"/>
1503     <element ref="sampl:Extensions" minOccurs="0"/>
1504     <element ref="sampl:Status"/>
1505   </sequence>
1506   <attribute name="ID" type="ID" use="required"/>
1507   <attribute name="InResponseTo" type="NCName" use="optional"/>
1508   <attribute name="Version" type="string" use="required"/>
1509   <attribute name="IssueInstant" type="dateTime" use="required"/>
1510   <attribute name="Destination" type="anyURI" use="optional"/>
1511 </complexType>

```

1512 3.2.2.1 Element <Status>

1513 The <Status> element contains the following elements:

1514 <StatusCode> [Required]

1515 A code representing the status of the corresponding request.

1516 <StatusMessage> [Optional]

1517 A message which MAY be returned to an operator.

1518 <StatusDetail> [Optional]

1519 Additional information concerning the status of the request.

1520 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1521 <element name="Status" type="sampl:StatusType"/>
1522 <complexType name="StatusType">
1523   <sequence>
1524     <element ref="sampl:StatusCode"/>
1525     <element ref="sampl:StatusMessage" minOccurs="0"/>
1526     <element ref="sampl:StatusDetail" minOccurs="0"/>
1527   </sequence>
1528 </complexType>

```

1529 3.2.2.2 Element <StatusCode>

1530 The <StatusCode> element specifies a code or a set of nested codes representing the status of the
1531 corresponding request. The <StatusCode> element has the following element and attribute:

1532 Value [Required]

1533 The status code value. This attribute contains a URI reference. The value of the topmost
1534 <StatusCode> element MUST be from the top-level list provided in this section.

1535 <StatusCode> [Optional]

1536 A subordinate status code that provides more specific information on an error condition.

1537 The permissible top-level <StatusCode> values are as follows:

1538 urn:oasis:names:tc:SAML:2.0:status:Success

1539 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1540 <StatusDetail> elements.

1541 urn:oasis:names:tc:SAML:2.0:status:Requester

1542 The request could not be performed due to an error on the part of the requester.

1543 urn:oasis:names:tc:SAML:2.0:status:Responder
1544 The request could not be performed due to an error on the part of the SAML responder or SAML
1545 authority.

1546 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch
1547 The SAML responder could not process the request because the version of the request message was
1548 incorrect.

1549 The following second-level status codes are referenced at various places in this specification. Additional
1550 second-level status codes MAY be defined in future versions of the SAML specification. SAML system
1551 entities are free to define more specific status codes by defining appropriate URI references.

1552 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed
1553 The responding provider was unable to successfully authenticate the principal.

1554 urn:oasis:names:tc:SAML:2.0:status:ErrorAttrNameOrValue
1555 Unexpected or invalid content was encountered within a <saml:Attribute> or
1556 <saml:AttributeValue> element.

1557 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy
1558 The responding provider does not support the specified name identifier format for the requested
1559 subject.

1560 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext
1561 The specified authentication context requirements cannot be met by the responder.

1562 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP
1563 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an
1564 <IDPList> can be resolved or that none of the supported identity providers are available.

1565 urn:oasis:names:tc:SAML:2.0:status:NoPassive
1566 Indicates the identity provider cannot authenticate the principal passively, as has been requested.

1567 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
1568 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1569 supported by the intermediary.

1570 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
1571 Indicates that an identity provider cannot authenticate the principal directly and is not permitted to
1572 proxy the request further.

1573 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
1574 The SAML responder or SAML authority is able to process the request but has chosen not to respond.
1575 This status code MAY be used when there is concern about the security context of the request
1576 message or the sequence of request messages received from a particular requester.

1577 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
1578 The SAML responder or SAML authority does not support the request.

1579 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
1580 The SAML responder cannot process any requests with the protocol version specified in the request.

1581 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
1582 The SAML responder cannot process the request because the protocol version specified in the
1583 request message is a major upgrade from the highest protocol version supported by the responder.

1584 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
1585 The SAML responder cannot process the request because the protocol version specified in the
1586 request message is too low.

1587 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
1588 The resource value provided in the request message is invalid or unrecognized.

1589 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
1590 The response message would contain more elements than the SAML responder is able to return.

1591 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1592 An entity has no knowledge of an attribute profile but is presented with an attribute drawn from a
1593 particular profile.

1594 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
1595 The responding provider does not recognize the principal specified or implied by the request.

1596 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
1597 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
1598 request.

1599 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
1600 type:

```
1601 <element name="StatusCode" type="samlp:StatusCodeType"/>
1602 <complexType name="StatusCodeType">
1603   <sequence>
1604     <element ref="samlp:StatusCode" minOccurs="0"/>
1605   </sequence>
1606   <attribute name="Value" type="anyURI" use="required"/>
1607 </complexType>
```

1608 **3.2.2.3 Element <StatusMessage>**

1609 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1610 The following schema fragment defines the <StatusMessage> element:

```
1611 <element name="StatusMessage" type="string"/>
```


1612 3.2.2.4 Element <StatusDetail>

1613 The <StatusDetail> element MAY be used to specify additional information concerning the status of
1614 the request. The additional information consists of zero or more elements from any namespace, with no
1615 requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

1616 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1617 complex type:

```
1618 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1619 <complexType name="StatusDetailType">  
1620 <sequence>  
1621 <any namespace="##any" processContents="lax" minOccurs="0"  
1622 maxOccurs="unbounded"/>  
1623 </sequence>  
1624 </complexType>
```

1625 3.3 Assertion Query and Request Protocol

1626 This section defines messages and processing rules for requesting existing assertions by reference or
1627 querying for assertions by subject and statement type.

1628 3.3.1 Element <AssertionIDRequest>

1629 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>
1630 message element can be used to request that they be returned in a <Response> message. The
1631 <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for
1632 more information on this element.

1633 The following schema fragment defines the <AssertionIDRequest> element:

```
1634 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>  
1635 <complexType name="AssertionIDRequestType">  
1636 <complexContent>  
1637 <extension base="samlp:RequestAbstractType">  
1638 <sequence>  
1639 <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>  
1640 </sequence>  
1641 </extension>  
1642 </complexContent>  
1643 </complexType>
```

1644 3.3.2 Queries

1645 The following sections define the SAML query request messages.

1646 3.3.2.1 Element <SubjectQuery>

1647 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1648 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and
1649 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the
1650 <saml:Subject> element to **RequestAbstractType**.

1651 The following schema fragment defines the <SubjectQuery> element and its
1652 **SubjectQueryAbstractType** complex type:

```
1653 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1654 <complexType name="SubjectQueryAbstractType" abstract="true">
```

```

1655     <complexContent>
1656         <extension base="saml:RequestAbstractType">
1657             <sequence>
1658                 <element ref="saml:Subject"/>
1659             </sequence>
1660         </extension>
1661     </complexContent>
1662 </complexType>

```

1663 3.3.2.2 Element <AuthnQuery>

1664 The <AuthnQuery> message element is used to make the query "What assertions containing
1665 authentication statements are available for this subject?" A successful <Response> will contain one or
1666 more assertions containing authentication statements.

1667 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1668 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1669 that have occurred in a previous interaction between the indicated subject and the authentication authority.

1670 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1671 the following element and attribute:

1672 **SessionIndex** [Optional]

1673 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1674 containing authentication statements do you have for this subject within the context of the supplied
1675 session information?"

1676 <RequestedAuthnContext> [Optional]

1677 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1678 containing authentication statements do you have for this subject that satisfy the authentication
1679 context requirements in this element?"

1680 In response to an authentication query, a SAML authority returns assertions with authentication
1681 statements as follows:

- 1682 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1683 assertions that may be returned.
- 1684 • If the **SessionIndex** attribute is present in the query, at least one <AuthnStatement> element in
1685 the set of returned assertions MUST contain a **SessionIndex** attribute that matches the
1686 **SessionIndex** attribute in the query. It is OPTIONAL for the complete set of all such matching
1687 assertions to be returned in the response.
- 1688 • If the <RequestedAuthnContext> element is present in the query, at least one
1689 <AuthnStatement> element in the set of returned assertions MUST contain an
1690 <AuthnContext> element that satisfies the element in the query (see Section 3.3.2.3). It is
1691 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1692 The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex
1693 type:

```

1694 <element name="AuthnQuery" type="saml:AuthnQueryType"/>
1695 <complexType name="AuthnQueryType">
1696     <complexContent>
1697         <extension base="saml:SubjectQueryAbstractType">
1698             <sequence>
1699                 <element ref="saml:RequestedAuthnContext" minOccurs="0"/>
1700             </sequence>
1701             <attribute name="SessionIndex" type="string" use="optional"/>
1702         </extension>
1703     </complexContent>

```

1704 </complexType>

1705 3.3.2.3 Element <RequestedAuthnContext>

1706 The <RequestedAuthnContext> element specifies the authentication context requirements of
1707 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
1708 complex type defines the following elements and attributes:

1709 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1710 Specifies one or more URI references identifying authentication context classes or declarations. (For
1711 more information about authentication context classes, see [SAMLAuthnCxt].)

1712 Comparison [Optional]

1713 Specifies the comparison method used to evaluate the requested context classes or statements, one
1714 of "exact", "minimum", "maximum", or "better". The default is "exact".

1715 Either a set of class references or a set of declaration references can be used. The set of supplied
1716 references MUST be evaluated as an ordered set, where the first element is the most preferred
1717 authentication context class or declaration. If none of the specified classes or declarations can be satisfied
1718 in accordance with the rules below, then the responder MUST return a <Response> message with a
1719 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1720 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
1721 statement MUST be the exact match of at least one of the authentication contexts specified.

1722 If Comparison is set to "minimum", then the resulting authentication context in the authentication
1723 statement MUST be at least as strong (as deemed by the responder) as one of the authentication
1724 contexts specified.

1725 If Comparison is set to "better", then the resulting authentication context in the authentication statement
1726 MUST be stronger (as deemed by the responder) than any one of the authentication contexts specified.

1727 If Comparison is set to "maximum", then the resulting authentication context in the authentication
1728 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength
1729 of at least one of the authentication contexts specified.

1730 The following schema fragment defines the <RequestedAuthnContext> element and its
1731 **RequestedAuthnContextType** complex type:

```
1732 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1733 <complexType name="RequestedAuthnContextType">
1734   <choice>
1735     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1736     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1737   </choice>
1738   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1739 use="optional"/>
1740 </complexType>
1741 <simpleType name="AuthnContextComparisonType">
1742   <restriction base="string">
1743     <enumeration value="exact"/>
1744     <enumeration value="minimum"/>
1745     <enumeration value="maximum"/>
1746     <enumeration value="better"/>
1747   </restriction>
1748 </simpleType>
```

1749 3.3.2.4 Element <AttributeQuery>

1750 The <AttributeQuery> element is used to make the query “Return the requested attributes for this
1751 subject.” A successful response will be in the form of assertions containing attribute statements, to the
1752 extent allowed by policy. This element is of type **AttributeQueryType**, which extends
1753 **SubjectQueryAbstractType** with the addition of the following element and attribute:

1754 <saml:Attribute> [Any Number]

1755 Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no
1756 attributes are specified, it indicates that all attributes allowed by policy are requested. If a given
1757 <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if
1758 that attribute is returned in the response, it **MUST NOT** contain any values that are not equal to the
1759 values specified in the query. In the absence of equality rules specified by particular profiles or
1760 attributes, equality is defined as an identical XML representation of the value.

1761 A single query **MUST NOT** contain two <saml:Attribute> elements with the same Name and
1762 NameFormat values (that is, a given attribute **MUST** be named only once in a query).

1763 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1764 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1765 assertions that may be returned.
- 1766 • If any <Attribute> elements are present in the query, they constrain/filter the attributes and
1767 optionally the values returned, as noted above.
- 1768 • The attributes and values returned **MAY** also be constrained by application-specific policy
1769 considerations.

1770 The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1771 and urn:oasis:names:tc:SAML:2.0:status:ErrorAttrNameOrValue **MAY** be used to indicate
1772 problems with the interpretation of attribute or value information in a query.

1773 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1774 complex type:

```
1775 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>  
1776 <complexType name="AttributeQueryType">  
1777   <complexContent>  
1778     <extension base="samlp:SubjectQueryAbstractType">  
1779       <sequence>  
1780         <element ref="saml:Attribute" minOccurs="0"  
1781         maxOccurs="unbounded"/>  
1782       </sequence>  
1783     </extension>  
1784   </complexContent>  
1785 </complexType>
```

1786 3.3.2.5 Element <AuthzDecisionQuery>

1787 The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource
1788 be allowed for this subject, given this evidence?” A successful response will be in the form of assertions
1789 containing authorization decision statements.

1790 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no
1791 future enhancements planned. Users who require additional functionality may want to
1792 consider the eXtensible Access Control Markup Language [XACML], which offers
1793 enhanced authorization decision features.

1794 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1795 addition of the following elements and attribute:

1796 Resource [Required]

1797 A URI reference indicating the resource for which authorization is requested.

1798 <saml:Action> [One or More]

1799 The actions for which authorization is requested.

1800 <saml:Evidence> [Optional]

1801 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1802 In response to an authorization decision query, a SAML authority returns assertions with authorization
1803 decision statements as follows:

- 1804 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1805 assertions that may be returned.

1806 The following schema fragment defines the <AuthzDecisionQuery> element and its
1807 **AuthzDecisionQueryType** complex type:

```
1808 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
1809 <complexType name="AuthzDecisionQueryType">
1810   <complexContent>
1811     <extension base="samlp:SubjectQueryAbstractType">
1812       <sequence>
1813         <element ref="saml:Action" maxOccurs="unbounded"/>
1814         <element ref="saml:Evidence" minOccurs="0"/>
1815       </sequence>
1816       <attribute name="Resource" type="anyURI" use="required"/>
1817     </extension>
1818   </complexContent>
1819 </complexType>
```

1820 3.3.3 Element <Response>

1821 The <Response> message element is used when a response consists of a list of zero or more assertions
1822 that answer the request. It has the complex type **ResponseType**, which extends **StatusResponseType**
1823 and adds the following elements:

1824 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1825 Specifies an assertion by value, or optionally an encrypted assertion by value. (See Section 2.3.3 for
1826 more information.)

1827 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1828 <element name="Response" type="samlp:ResponseType"/>
1829 <complexType name="ResponseType">
1830   <complexContent>
1831     <extension base="samlp:StatusResponseType">
1832       <choice minOccurs="0" maxOccurs="unbounded">
1833         <element ref="saml:Assertion"/>
1834         <element ref="saml:EncryptedAssertion"/>
1835       </choice>
1836     </extension>
1837   </complexContent>
1838 </complexType>
```

1839 3.3.4 Processing Rules

1840 In response to a query message, every assertion returned by a SAML authority **MUST** contain a
1841 `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the query.

1842 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both
1843 apply:

- 1844 • If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST**
1845 include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2.
1846 In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical"
1847 means that the identifier element's content and attribute values **MUST** be the same. An encrypted
1848 identifier will be identical to the original according to this definition, once decrypted.
- 1849 • If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at
1850 least one `<saml:SubjectConfirmation>` element such that the assertion's subject can be
1851 confirmed in the manner described by at least one element in the requested set.

1852 As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular
1853 `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting
1854 S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with
1855 different `Format` values and element content, even if the two identifiers are considered to refer to the
1856 same principal.

1857 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
1858 expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an
1859 `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value
1860 `urn:oasis:names:tc:SAML:2.0:status:Success`.

1861 All other processing rules associated with the underlying request and response messages **MUST** be
1862 observed.

1863 3.4 Authentication Request Protocol

1864 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
1865 authentication statements to establish a security context at one or more relying parties, it can use the
1866 authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and
1867 request that it return a `<Response>` message containing one or more such assertions. Such assertions
1868 **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one
1869 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

1870 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
1871 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
1872 are not specified, though the means of authentication might impact the content of the response. Other
1873 issues related to the validation of authentication credentials by the identity provider or any communication
1874 between the identity provider and any other entities involved in the authentication process are also out of
1875 scope of this protocol.

1876 The descriptions and processing rules in the following sections reference the following actors, many of
1877 whom might be the same entity in a particular profile of use:

1878 Request Issuer

1879 The entity who creates the authentication request and to whom the response is to be returned.

1880 Presenter

1881 The entity who presents the request to the authority and either authenticates itself during the
1882 sending of the message, or relies on an existing security context to establish its identity. If not the

1883 request issuer, the sender acts as an intermediary between the request issuer and the responding
1884 identity provider.

1885 Requested Subject

1886 The entity about whom one or more assertions are being requested.

1887 Confirming Subject

1888 The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>
1889 elements of the resulting assertion(s).

1890 Relying Party

1891 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
1892 the profile or context of use, generally to establish a security context.

1893 3.4.1 Element <AuthnRequest>

1894 To request that an identity provider issue an assertion with an authentication statement, a presenter
1895 authenticates to that identity provider (or relies on an existing security context) and sends it an
1896 <AuthnRequest> message that describes the properties that the resulting assertion needs to have to
1897 satisfy its purpose. Among these properties may be information that relates to the content of the assertion
1898 and/or information that relates to how the resulting <Response> message should be delivered to the
1899 request issuer. The process of authentication of the presenter may take place before, during, or after the
1900 initial delivery of the <AuthnRequest> message.

1901 The request issuer might not be the same as the presenter of the request, if for example the request
1902 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the
1903 requested subject to provide a service.

1904 The <AuthnRequest> message SHOULD be signed or otherwise authenticated and integrity protected
1905 by the protocol binding used to deliver the message.

1906 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
1907 adds the following elements and attributes, all of which are optional in general, but may be required by
1908 specific profiles:

1909 <saml:Subject> [Optional]

1910 Specifies the requested subject of the resulting assertion(s). This may include one or more
1911 <saml:SubjectConfirmation> elements to indicate how and/or by whom the resulting assertions
1912 can be confirmed.

1913 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
1914 requested subject. If no <saml:SubjectConfirmation> elements are included, then the presenter
1915 is presumed to be the only confirming entity required and the method is implied by the profile of use
1916 and/or the policies of the identity provider.

1917 <NameIDPolicy> [Optional]

1918 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
1919 then any type of identifier supported by the identity provider for the requested subject can be used,
1920 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

1921 <saml:Conditions> [Optional]

1922 Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the
1923 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
1924 information in this element is used as input to the process of constructing the assertion, rather than as
1925 conditions on the use of the request itself.

- 1926 <RequestedAuthnContext> [Optional]
1927 Specifies the requirements, if any, that the request issuer places on the authentication context that
1928 applies to the responding provider's authentication of the presenter. See Section 3.3.2.3 for
1929 processing rules regarding this element.
- 1930 <Scoping> [Optional]
1931 Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as
1932 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
1933 providers by the responder.
- 1934 IsPassive [Optional]
1935 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of the
1936 user interface from the request issuer and interact with the presenter in a noticeable fashion. If a value
1937 is not provided, the default is "false".
- 1938 ForceAuthn [Optional]
1939 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
1940 rely on a previous security context. If a value is not provided, the default is "false". However, if both
1941 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
1942 presenter unless the constraints of IsPassive can be met.
- 1943 ProtocolBinding [Optional]
1944 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>
1945 message. See [SAMLBind] for more information about protocol bindings and URI references defined
1946 for them.
- 1947 AssertionConsumerServiceIndex [Optional]
1948 Indirectly identifies the location to which the <Response> message should be returned to the request
1949 issuer. It applies only to profiles in which the request issuer is different from the presenter. The identity
1950 provider MUST have a trusted means to map the index value in the attribute to a location associated
1951 with the request issuer. [SAMLMeta] provides one possible mechanism. If omitted, then the identity
1952 provider MUST return the <Response> message to the default location associated with the request
1953 issuer for the profile of use. This attribute is mutually exclusive with the
1954 AssertionConsumerServiceURL attribute.
- 1955 AssertionConsumerServiceURL [Optional]
1956 Specifies by value the location to which the <Response> message MUST be returned to the request
1957 issuer. The responder MUST ensure by some means that the value specified is in fact associated with
1958 the request issuer. [SAMLMeta] provides one possible mechanism; signing the enclosing
1959 <AuthnRequest> message is another.. This attribute is mutually exclusive with the
1960 AssertionConsumerServiceIndex attribute.
- 1961 AttributeConsumingServiceIndex [Optional]
1962 Indirectly identifies information associated with the request issuer describing the SAML attributes the
1963 request issuer desires or requires be supplied by the identity provider in the <Response> message.
1964 The identity provider MUST have a trusted means to map the index value in the attribute to
1965 information associated with the request issuer. [SAMLMeta] provides one possible mechanism. The
1966 identity provider MAY use this information to populate one or more <saml:AttributeStatement>
1967 elements in the assertion(s) it returns.
- 1968 ProviderName [Optional]
1969 Specifies the human-readable name of the request issuer for use by the presenter's user agent or the
1970 identity provider.
- 1971 See Section 3.4.1.4 for general processing rules regarding this message.

1972 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**
1973 complex type:

```
1974 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
1975 <complexType name="AuthnRequestType">
1976   <complexContent>
1977     <extension base="samlp:RequestAbstractType">
1978       <sequence>
1979         <element ref="saml:Subject" minOccurs="0"/>
1980         <element ref="saml:NameIDPolicy" minOccurs="0"/>
1981         <element ref="saml:Conditions" minOccurs="0"/>
1982         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
1983         <element ref="samlp:Scoping" minOccurs="0"/>
1984       </sequence>
1985       <attribute name="IsPassive" type="boolean" use="optional"/>
1986       <attribute name="ForceAuthn" type="boolean" use="optional"/>
1987       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
1988       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
1989 use="optional"/>
1990       <attribute name="AssertionConsumerServiceURL" type="anyURI"
1991 use="optional"/>
1992       <attribute name="AttributeConsumingServiceIndex"
1993 type="unsignedShort" use="optional"/>
1994       <attribute name="ProviderName" type="string" use="optional"/>
1995     </extension>
1996   </complexContent>
1997 </complexType>
```

1998 3.4.1.1 Element <NameIDPolicy>

1999 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
2000 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

2001 Format [Required]

2002 Specifies the URI reference corresponding to a name identifier format defined in this or another
2003 specification (see Section 8.3 for examples). The additional value of
2004 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use
2005 within this attribute to indicate a request that the resulting identifier be encrypted.

2006 SPNameQualifier [Optional]

2007 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of
2008 a service provider other than the request issuer, or in the namespace of an affiliation group of service
2009 providers. See for example the definition of urn:oasis:names:tc:SAML:2.0:nameid-
2010 format:persistent in section 8.3.7.

2011 AllowCreate [Optional]

2012 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the
2013 request, to create a new identifier to represent the principal. Defaults to "false". When "false", the
2014 request issuer constrains the identity provider to only issue an assertion to it if an acceptable identifier
2015 for the principal has already been established.

2016 When this element is used, if the content is not understood by or acceptable to the identity provider, then
2017 a <Response> message element MUST be returned with an error <Status>, and MAY contain a
2018 second-level <StatusCode> of
2019 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy.

2020 The special Format value urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates
2021 that the resulting assertion(s) MUST contain <EncryptedID> elements instead of plaintext. The
2022 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the
2023 requested subject.

2024 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an
2025 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly
2026 specific to the service provider) require that an encrypted identifier be used.

2027 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
2028 complex type:

```
2029 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
2030 <complexType name="NameIDPolicyType">
2031   <attribute name="Format" type="anyURI" use="required"/>
2032   <attribute name="SPNameQualifier" type="string" use="optional"/>
2033   <attribute name="AllowCreate" type="boolean" use="optional"/>
2034 </complexType>
```

2035 3.4.1.2 Element `<Scoping>`

2036 The `<Scoping>` element specifies the identity providers trusted by the request issuer to authenticate the
2037 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
2038 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
2039 elements and attribute:

2040 `ProxyCount` [Optional]

2041 Specifies the number of proxying indirections permissible between the identity provider that receives
2042 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of
2043 zero permits no proxying, while omitting this attribute expresses no such restriction.

2044 `<IDPList>` [Optional]

2045 An advisory list of identity providers and associated information that the request issuer deems
2046 acceptable to respond to the request.

2047 `<RequesterID>` [Zero or More]

2048 Identifies the set of requesting entities on whose behalf the request issuer is acting. Used to
2049 communicate the chain of request issuers when proxying occurs, as described in Section 3.4.1.5. See
2050 Section 8.3.6 for a description of entity identifiers.

2051 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
2052 `<Response>` message with an error `<Status>` and a second-level `<StatusCode>` of
2053 `urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP` or
2054 `urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP` if it cannot contact or does not support
2055 any of the specified identity providers.

2056 The following schema fragment defines the `<Scoping>` element and its **ScopingType** complex type:

```
2057 <element name="Scoping" type="samlp:ScopingType"/>
2058 <complexType name="ScopingType">
2059   <sequence>
2060     <element ref="samlp:IDPList" minOccurs="0"/>
2061     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2062   </sequence>
2063   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2064 </complexType>
2065 <element name="RequesterID" type="anyURI"/>
```

2066 3.4.1.3 Element `<IDPList>`

2067 The `<IDPList>` element specifies the identity providers trusted by the request issuer to authenticate the
2068 presenter. Its **IDPListType** complex type defines the following elements:

2069 <IDPEntry> [One or More]
2070 Information about a single identity provider.
2071 <GetComplete> [Optional]
2072 If the <IDPList> is not complete, using this element specifies a URI reference that resolves to the
2073 complete list.

2074 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
2075 <element name="IDPList" type="samlp:IDPListType"/>  
2076 <complexType name="IDPListType">  
2077   <sequence>  
2078     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>  
2079     <element ref="samlp:GetComplete" minOccurs="0"/>  
2080   </sequence>  
2081 </complexType>  
2082 <element name="GetComplete" type="anyURI"/>
```

2083 3.4.1.3.1 Element <IDPEntry>

2084 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to authenticate
2085 the presenter. Its **IDPEntryType** complex type defines the following attributes:

2086 ProviderID [Required]

2087 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2088 Name [Optional]

2089 A human-readable name for the identity provider.

2090 Loc [Optional]

2091 A URI reference representing the location of a profile-specific endpoint supporting the authentication
2092 request protocol. The binding to be used must be understood from the profile of use.

2093 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
2094 <element name="IDPEntry" type="samlp:IDPEntryType"/>  
2095 <complexType name="IDPEntryType">  
2096   <attribute name="ProviderID" type="anyURI" use="required"/>  
2097   <attribute name="Name" type="string" use="optional"/>  
2098   <attribute name="Loc" type="anyURI" use="optional"/>  
2099 </complexType>
```

2100 3.4.1.4 Processing Rules

2101 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
2102 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
2103 kinds of input and output are required or prohibited. The following processing rules apply as invariant
2104 behavior across any profile of this protocol exchange. All other processing rules associated with the
2105 underlying request and response messages **MUST** also be observed.

2106 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing
2107 one or more assertions that meet the specifications defined by the request, or with a <Response>
2108 message containing a <Status> describing the error that occurred. The responder **MAY** conduct
2109 additional message exchanges with the presenter as needed to initiate or complete the authentication
2110 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
2111 the next section, this includes proxying the request by directing the presenter to another identity provider
2112 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to

2113 authenticate the presenter to the original responder, in effect using SAML as the authentication
2114 mechanism.

2115 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if
2116 prevented from providing an assertion by policies in effect at the identity provider (for example the
2117 intended subject has prohibited the identity provider from providing assertions to the relying party), then it
2118 **MUST** return a <Response> with an error <Status>, and **MAY** return a second-level <StatusCode> of
2119 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
2120 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2121 If the <saml:Subject> element in the request is present, then the resulting assertions'
2122 <saml:Subject> **MUST strongly match** the request <saml:Subject>, as described in Section 3.3.4,
2123 **except** that the identifier **MAY** be in a different format if specified by <NameIDPolicy>. In such a case,
2124 the identifier's physical content **MAY** be different, but it **MUST** refer to the same principal.

2125 All of the content defined specifically within <AuthnRequest> is optional, although some may be required
2126 by certain profiles. In the absence of any specific content at all, the following behavior is assumed:

- 2127 • The assertion(s) returned **MUST** contain a <saml:Subject> element that represents the
2128 presenter. The identifier type and format are determined by the identity provider. At least one
2129 statement in at least one assertion **MUST** be a <saml:AuthnStatement> that describes the
2130 authentication performed by the responder or authentication service associated with it.
- 2131 • The request presenter should, to the extent possible, be the only entity able to satisfy the
2132 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation
2133 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- 2134 • The resulting assertion(s) **MUST** contain a <saml:AudienceRestriction> element
2135 referencing the request issuer as an acceptable relying party. Other audiences **MAY** be included as
2136 deemed appropriate by the identity provider.

2137 **3.4.1.5 Proxying**

2138 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or
2139 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
2140 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
2141 <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in
2142 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
2143 identity provider.

2144 Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the
2145 enclosed assertion or non-SAML equivalent **MAY** be used to authenticate the presenter so that the
2146 proxying provider can issue an assertion of its own in response to the original <AuthnRequest>,
2147 completing the overall message exchange. Both the proxying and authenticating identity providers **MAY**
2148 include constraints on proxying activity in the messages and assertions they issue, as described in
2149 previous sections and below.

2150 The request issuer can influence proxy behavior by including a <Scoping> element where the provider
2151 sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be
2152 proxied by including an ordered <IDPList> of preferred providers.

2153 An identity provider can control secondary use of its assertions by proxying identity providers using a
2154 <ProxyRestriction> element in the assertions it issues.

2155 3.4.1.5.1 Proxying Processing Rules

2156 An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is
2157 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2158 choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

2159 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity
2160 provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of
2161 `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly
2162 authenticate the presenter.

2163 If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying
2164 identity provider MUST include equivalent or stricter forms of all the information included in the original
2165 request (such as authentication context policy). Note, however, that the proxying provider is free to specify
2166 whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

2167 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
2168 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for
2169 example) will be honored by the authenticating provider.

2170 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less
2171 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new
2172 request SHOULD contain a `<ProxyCount>` attribute.

2173 If an `<IDPList>` was specified in the original request, the new request MUST also contain an
2174 `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the
2175 `<IDPList>`, but MUST NOT remove any from the list.

2176 The authentication request and response are processed in normal fashion, in accordance with the rules
2177 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
2178 provider (in the case of SAML by delivering a `<Response>`), the following steps are followed:

- 2179 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
2180 relevant information from the original assertion or non-SAML equivalent.
- 2181 • The new assertion's `<saml:Subject>` MUST contain an identifier that satisfies the original
2182 request issuer's preferences, as defined by its `<NameIDPolicy>` element.
- 2183 • The `<saml:AuthnStatement>` in the new assertion MUST include a `<saml:AuthnContext>`
2184 element containing a `<saml:AuthenticatingAuthority>` element referencing the identity
2185 provider to which the proxying identity provider referred the presenter. If the original assertion
2186 contains `<saml:AuthnContext>` information that includes one or more
2187 `<saml:AuthenticatingAuthority>` elements, those elements SHOULD be included in the
2188 new assertion, with the new element placed after them.
- 2189 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
2190 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
2191 consistent over time across different requests. The value MUST not conflict with values used or
2192 generated by other SAML providers.
- 2193 • Any other `<saml:AuthnContext>` information MAY be copied, translated, or omitted in
2194 accordance with the policies of the proxying identity provider, provided that the original
2195 requirements dictated by the request issuer are met.

2196 If, in the future, the identity provider is asked to authenticate the same presenter for a second request
2197 issuer, and this request is equally or less strict than the original request (as determined by the proxying
2198 identity provider), the identity provider MAY skip the creation of a new `<AuthnRequest>` to the
2199 authenticating identity provider and immediately issue another assertion (assuming the original assertion
2200 or non-SAML equivalent it received is still valid).

2201 3.5 Artifact Resolution Protocol

2202 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
2203 transported in a SAML binding by reference instead of by value. Both requests and responses can be
2204 obtained by reference using this specialized protocol. A message sender, instead of binding a message to
2205 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2206 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver
2207 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
2208 protocol to resolve the artifact into the original protocol message.

2209 The most common use for this mechanism is with bindings that cannot easily carry a message because of
2210 size constraints, or to enable a message to be communicated via a secure channel between the SAML
2211 requester and responder, avoiding the need for a signature.

2212 Depending on the characteristics of the underlying message being passed by reference, the artifact
2213 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2214 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
2215 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
2216 by any party.

2217 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
2218 as if the message so obtained had been sent originally in place of the artifact.

2219 3.5.1 Element <ArtifactResolve>

2220 The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an
2221 <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message.
2222 The original transmission of the artifact is governed by the specific protocol binding that is being used; see
2223 [SAMLBind] for more information on the use of artifacts in bindings.

2224 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
2225 protected by the protocol binding used to deliver the message.

2226 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
2227 adds the following element:

2228 <Artifact> [Required]

2229 The artifact value that the requester received and now wishes to translate into the protocol message it
2230 represents. See [SAMLBind] for specific artifact format information.

2231 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**
2232 complex type:

```
2233 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
2234 <complexType name="ArtifactResolveType">
2235   <complexContent>
2236     <extension base="samlp:RequestAbstractType">
2237       <sequence>
2238         <element ref="samlp:Artifact"/>
2239       </sequence>
2240     </extension>
2241   </complexContent>
2242 </complexType>
2243 <element name="Artifact" type="string"/>
```

2244 3.5.2 Element <ArtifactResponse>

2245 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
2246 message element. This element is of complex type **ArtifactResponseType**, which extends
2247 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2248 message being returned. This wrapped message element can be a request or a response.

2249 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2250 protected by the protocol binding used to deliver the message.

2251 The following schema fragment defines the <ArtifactResponse> element and its
2252 **ArtifactResponseType** complex type:

```
2253 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>  
2254 <complexType name="ArtifactResponseType">  
2255   <complexContent>  
2256     <extension base="samlp:StatusResponseType">  
2257       <sequence>  
2258         <any namespace="##any" processContents="lax" minOccurs="0"/>  
2259       </sequence>  
2260     </extension>  
2261   </complexContent>  
2262 </complexType>
```

2263 3.5.3 Processing Rules

2264 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2265 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>
2266 element with no embedded message. In both cases, the <Status> element MUST include a
2267 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A
2268 response message with no embedded message inside it is termed an empty response in the remainder of
2269 this section.

2270 The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent
2271 request with the same artifact by any requester results in an empty response as described above.

2272 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY
2273 be intended for consumption by any party that receives it and can respond appropriately. In most other
2274 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2275 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2276 receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the
2277 original intended recipient, then the artifact issuer MUST return an empty response.

2278 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2279 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2280 return it in an <ArtifactResolve> message to the issuer.

2281 Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of
2282 the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message
2283 will contain its own message identifier, and in the case of an embedded response, may contain a different
2284 InResponseTo value that corresponds to the original request message to which the embedded message
2285 is responding.

2286 All other processing rules associated with the underlying request and response messages MUST be
2287 observed.

2288 3.6 Name Identifier Management Protocol

2289 After establishing a persistent name identifier for a principal, an identity provider wishing to change the
2290 value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name
2291 identifier will no longer be used to refer to the principal, informs service providers of the change by
2292 sending them a <ManageNameIDRequest> message.

2293 A service provider also uses this message to register or change the *SPProvidedID* value to be included
2294 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
2295 identifier between itself and the identity provider.

2296 3.6.1 Element <ManageNameIDRequest>

2297 A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name
2298 identifier or to indicate the termination of the use of a name identifier.

2299 The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity
2300 protected by the protocol binding used to deliver the message.

2301 This message has the complex type **ManageNameIDRequestType**, which extends
2302 **RequestAbstractType** and adds the following elements:

2303 <saml:NameID> or <saml:EncryptedID> [Required]

2304 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
2305 principal as currently recognized by the identity and service providers prior to this request.

2306 <NewID> or <NewEncryptedID> or <Terminate> [Required]

2307 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2308 requesting provider concerning this principal, or an indication that the use of the old identifier has
2309 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
2310 appear in subsequent <NameID> elements in the *SPProvidedID* attribute. If the requester is the
2311 identity provider, the new value will appear in subsequent <NameID> elements as the element's
2312 content.

2313 The following schema fragment defines the <ManageNameIDRequest> element and its
2314 **ManageNameIDRequestType** complex type:

```
2315 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2316 <complexType name="ManageNameIDRequestType">
2317   <complexContent>
2318     <extension base="samlp:RequestAbstractType">
2319       <sequence>
2320         <choice>
2321           <element ref="saml:NameID"/>
2322           <element ref="saml:EncryptedID"/>
2323         </choice>
2324         <choice>
2325           <element ref="samlp:NewID"/>
2326           <element ref="samlp:NewEncryptedID"/>
2327           <element ref="samlp:Terminate"/>
2328         </choice>
2329       </sequence>
2330     </extension>
2331   </complexContent>
2332 </complexType>
2333 <element name="NewID" type="string"/>
2334 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
2335 <element name="Terminate" type="samlp:TerminateType"/>
2336 <complexType name="TerminateType"/>
```


2337 3.6.2 Element <ManageNameIDResponse>

2338 The recipient of a <ManageNameIDRequest> message MUST respond with a
2339 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional
2340 content.

2341 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity
2342 protected by the protocol binding used to deliver the message.

2343 The following schema fragment defines the <ManageNameIDResponse> element:

```
2344 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

2345 3.6.3 Processing Rules

2346 If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize,
2347 the responding provider MUST respond with an error <Status> and MAY respond with a second-level
2348 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2349 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the
2350 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of
2351 an identity provider) it will no longer issue assertions to the service provider about the principal. The
2352 receiving provider can perform any maintenance with the knowledge that the relationship represented by
2353 the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for
2354 whom a relationship has been terminated.

2355 If the service provider requests that its identifier for the principal be changed by including a <NewID> (or
2356 <NewEncryptedID>) element, the identity provider MUST include the element's content as the
2357 SPProvidedID when subsequently communicating to the service provider regarding this principal.

2358 If the identity provider requests that its identifier for the principal be changed by including a <NewID> (or
2359 <NewEncryptedID>) element, the service provider MUST use the element's content as the
2360 <saml:NameID> element content when subsequently communicating with the identity provider regarding
2361 this principal.

2362 Note that neither, either, or both of the original and new identifier MAY be encrypted (using the
2363 <EncryptedID> and <NewEncryptedID> elements).

2364 In any case, the <saml:NameID> content in the request and its associated SPProvidedID attribute
2365 MUST contain the most recent name identifier information established between the providers for the
2366 principal.

2367 In the case of an identifier with a Format of urn:oasis:names:tc:SAML:2.0:nameid-
2368 format:persistent OR urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted, the
2369 NameQualifier attribute MUST contain the unique identifier of the identity provider or be omitted. If the
2370 identifier was established between the identity provider and an affiliation group of which the service
2371 provider is a member, then the SPNameQualifier attribute MUST contain the unique identifier of the
2372 affiliation group. Otherwise, it MUST contain the unique identifier of the service provider or be omitted.

2373 Changes to these identifiers may take a potentially significant amount of time to propagate through the
2374 systems at both the requester and the responder. Implementations might wish to allow each party to
2375 accept either identifier for some period of time following the successful completion of a name identifier
2376 change. Not doing so could result in the inability of the principal to access resources.

2377 All other processing rules associated with the underlying request and response messages MUST be
2378 observed.

2379 **3.7 Single Logout Protocol**

2380 The single logout protocol provides a message exchange protocol by which all sessions provided by a
2381 particular session authority are near-simultaneously terminated. The single logout protocol is used either
2382 when a principal logs out at a session participant or when the principal logs out directly at the
2383 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
2384 the logout event can be indicated through the `Reason` attribute.

2385
2386 The principal may have established authenticated sessions with both the session authority and individual
2387 session participants, based on assertions containing authentication statements supplied by the session
2388 authority.

2389
2390 When the principal invokes the single logout process at a session participant, the session participant
2391 MUST send a `<LogoutRequest>` message to the session authority that provided the assertion
2392 containing the authentication statement related to that session at the session participant.

2393
2394 When either the principal invokes a logout at the session authority, or a session participant sends a logout
2395 request to the session authority specifying that principal, the session authority MUST send a
2396 `<LogoutRequest>` message to each session participant to which it provided assertions containing
2397 authentication statements under its current session with the principal, with the exception of the session
2398 participant that sent the `<LogoutRequest>` message to the session authority.

2399 **3.7.1 Element `<LogoutRequest>`**

2400 A session participant or session authority sends a `<LogoutRequest>` message to indicate that a session
2401 has been terminated.

2402 The `<LogoutRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2403 by the protocol binding used to deliver the message.

2404 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and
2405 adds the following elements and attributes:

2406 `NotOnOrAfter` [Optional]

2407 The time at which the request expires, after which the recipient may discard the message. The time
2408 value is encoded in UTC, as described in Section 1.2.3.

2409 `Reason` [Optional]

2410 An indication of the reason for the logout, in the form of a URI reference.

2411 `<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

2412 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as
2413 currently recognized by the identity and service providers prior to this request.

2414 `<SessionIndex>` [Optional]

2415 The identifier that indexes this session at the message recipient.

2416 The following schema fragment defines the `<LogoutRequest>` element and associated
2417 **LogoutRequestType** complex type:

```
2418 <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
2419 <complexType name="LogoutRequestType">
2420   <complexContent>
2421     <extension base="samlp:RequestAbstractType">
2422       <sequence>
2423         <choice>
2424           <element ref="saml:BaseID"/>
```

```

2425         <element ref="saml:NameID"/>
2426         <element ref="saml:EncryptedID"/>
2427     </choice>
2428     <element ref="samlp:SessionIndex" minOccurs="0"
2429 maxOccurs="unbounded"/>
2430 </sequence>
2431     <attribute name="Reason" type="anyURI" minOccurs="0"/>
2432     <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
2433 </extension>
2434 </complexContent>
2435 </complexType>
2436 <element name="SessionIndex" type="string"/>

```

2437 3.7.2 Element <LogoutResponse>

2438 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of
2439 type **StatusResponseType**, with no additional content specified.

2440 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
2441 protected by the protocol binding used to deliver the message.

2442 The following schema fragment defines the <LogoutResponse> element:

```

2443 <element name="LogoutResponse" type="samlp:StatusResponseType"/>

```

2444 3.7.3 Processing Rules

2445 The message sender MAY use the `Reason` attribute to indicate the reason for sending the
2446 <LogoutRequest>. The following values are defined by this specification for use by all message
2447 senders; other values MAY be agreed on between participants:

2448 `urn:oasis:names:tc:SAML:2.0:logout:user`

2449 Specifies that the message is being sent because the principal wishes to terminate the indicated
2450 session.

2451 `urn:oasis:names:tc:SAML:2.0:logout:admin`

2452 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2453 session for that principal.

2454 All other processing rules associated with the underlying request and response messages MUST be
2455 observed.

2456 Additional processing rules are provided in the following sections.

2457 3.7.3.1 Session Participant Rules

2458 When a session participant receives a <LogoutRequest> message, the session participant MUST
2459 authenticate the message. If the sender is the authority that provided an assertion containing an
2460 authentication statement linked to the principal's current session, the session participant MUST invalidate
2461 the principal's session(s) referred to by the <saml:BaseID>, <saml:NameID>, or
2462 <saml:EncryptedID> element, and any <SessionIndex> elements supplied in the message. If no
2463 <SessionIndex> elements are supplied, then all sessions associated with the principal MUST be
2464 invalidated.

2465

2466 The session participant MUST apply the logout request message to any assertion that meets the following
2467 conditions, even if the assertion arrives after the logout request:

- 2468 • The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or
2469 `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in section 3.3.4.
- 2470 • The `SessionIndex` attribute of one of the assertion's authentication statements matches one of
2471 the `<SessionIndex>` elements specified in the logout request, or the logout request contains no
2472 `<SessionIndex>` elements.
- 2473 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself
2474 (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject
2475 confirmation data).
- 2476 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on
2477 the message).

2478 **Note:** This rule is intended to prevent a situation in which a session participant receives a
2479 logout request targeted at a single, or multiple, assertion(s) (as identified by the
2480 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid -
2481 assertion(s) targeted by the logout request. It should honor the logout request until the
2482 logout request itself may be discarded (the `NotOnOrAfter` value on the request has
2483 been exceeded) or the assertion targeted by the logout request has been received and
2484 has been handled appropriately.

2485 3.7.3.2 Session Authority Rules

2486 When a session authority receives a `<LogoutRequest>` message, the session authority **MUST**
2487 authenticate the sender. If the sender is a session participant to which the session authority provided an
2488 assertion containing an authentication statement for the current session, then the session authority
2489 **SHOULD** do the following in the specified order:

- 2490 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session
2491 authority proxied the user's authentication, unless the second authority is the originator of the
2492 `<LogoutRequest>`.
- 2493 • Send a `<LogoutRequest>` message to each session participant for which the session authority
2494 provided assertions in the current session, *other than* the originator of a current
2495 `<LogoutRequest>`.
- 2496 • Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`,
2497 or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout
2498 request message.

2499 If an error occurs during this further processing of the logout (for example, other session participants may
2500 not all implement the particular single logout protocol binding used by the requesting session participant),
2501 then the session authority **MUST** respond to the original requester with a `<LogoutResponse>` message,
2502 indicating the status of the logout request. The value
2503 `urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding` is provided for a second-level
2504 `<StatusCode>`, indicating that the originating session participant should retry the `<LogoutRequest>`
2505 using a different protocol binding.

2506 Note that this protocol does not permit "partial-success"; any error during the propagation of logout
2507 requests by the session authority **MUST** result in the failure of the overall logout process, and the session
2508 authority **MUST** return a `<LogoutResponse>` message containing an appropriate status code to the
2509 originating session participant, if any.

2510 Note that a session authority **MAY** initiate a logout for reasons other than having received a
2511 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 2512 • If some timeout period was agreed out-of-band with an individual session participant, the session
2513 authority **MAY** send a `<LogoutRequest>` to that individual participant alone.

- 2514 • An agreed global timeout period has been exceeded.
- 2515 • The principal or some other trusted entity has requested logout of the principal directly at the session
2516 authority.
- 2517 • The session authority has determined that the principal's credentials may have been compromised.

2518 When constructing a logout request message, the session authority MUST set the value of the
2519 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message,
2520 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time
2521 value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most
2522 recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout
2523 request).

2524 In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also
2525 available for use by the session authority only:

2526 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2527 Specifies that the message is being sent because of the global session timeout interval period
2528 being exceeded.

2529 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

2530 Specifies that the message is being sent because a timeout interval period agreed between a
2531 participant and the authority has been exceeded.

2532 **3.8 Name Identifier Mapping Protocol**

2533 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2534 identifier for the same principal in a particular format or federation namespace, it can send a request to
2535 the identity provider using this protocol.

2536 For example, a service provider that wishes to communicate with another service provider with whom it
2537 does not share an identifier for the principal can use an identity provider that shares an identifier for the
2538 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,
2539 with which it can communicate with the second service provider.

2540 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
2541 `<saml:EncryptedID>` element unless a specific deployment dictates such protection is unnecessary.

2542 **3.8.1 Element `<NameIDMappingRequest>`**

2543 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2544 `<NameIDMappingRequest>` message. This message has the complex type
2545 **`NameIDMappingRequestType`**, which extends **`RequestAbstractType`** and adds the following elements:

2546 `<saml:BaseID>` OR `<saml:NameID>` OR `<saml:EncryptedID>` [Required]

2547 The identifier and associated descriptive data that specify the principal as currently recognized by the
2548 requester and the responder.

2549 `<NameIDPolicy>` [Required]

2550 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2551 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2552 binding used to deliver the message.

2553 The following schema fragment defines the `<NameIDMappingRequest>` element and its
2554 **NameIDMappingRequestType** complex type:

```
2555 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2556 <complexType name="NameIDMappingRequestType">
2557   <complexContent>
2558     <extension base="samlp:RequestAbstractType">
2559       <sequence>
2560         <choice>
2561           <element ref="saml:BaseID"/>
2562           <element ref="saml:NameID"/>
2563           <element ref="saml:EncryptedID"/>
2564         </choice>
2565         <element ref="samlp:NameIDPolicy"/>
2566       </sequence>
2567     </extension>
2568   </complexContent>
2569 </complexType>
```

2570 3.8.2 Element `<NameIDMappingResponse>`

2571 The recipient of a `<NameIDMappingRequest>` message MUST respond with a
2572 `<NameIDMappingResponse>` message. This message has the complex type
2573 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

2574 `<saml:NameID>` or `<saml:EncryptedID>` [Required]

2575 The identifier and associated attributes that specify the principal in the manner requested, usually in
2576 encrypted form.

2577 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2578 binding used to deliver the message.

2579 The following schema fragment defines the `<NameIDMappingResponse>` element and its
2580 **NameIDMappingResponseType** complex type:

```
2581 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2582 <complexType name="NameIDMappingResponseType">
2583   <complexContent>
2584     <extension base="samlp:StatusResponseType">
2585       <choice>
2586         <element ref="saml:NameID"/>
2587         <element ref="saml:EncryptedID"/>
2588       </choice>
2589     </extension>
2590   </complexContent>
2591 </complexType>
```

2592 3.8.3 Processing Rules

2593 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2594 `<Status>` containing a second-level `<StatusCode>` of
2595 `urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`.

2596 At the responder's discretion, the
2597 `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy` status code MAY be returned to
2598 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2599 All other processing rules associated with the underlying request and response messages MUST be
2600 observed.

2601 4 SAML Versioning

2602 The SAML specification set is versioned in two independent ways. Each is discussed in the following
2603 sections, along with processing rules for detecting and handling version differences. Also included are
2604 guidelines on when and why specific version information is expected to change in future revisions of the
2605 specification.

2606 When version information is expressed as both a Major and Minor version, it is expressed in the form
2607 *Major.Minor*. The version number *Major_B.Minor_B* is higher than the version number *Major_A.Minor_A* if and
2608 only if:

2609 $Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B > Minor_A)$

2610 4.1 SAML Specification Set Version

2611 Each release of the SAML specification set will contain a major and minor version designation describing
2612 its relationship to earlier and later versions of the specification set. The version will be expressed in the
2613 content and filenames of published materials, including the specification set documents and XML schema
2614 documents. There are no normative processing rules surrounding specification set versioning, since it
2615 merely encompasses the collective release of normative specification documents which themselves
2616 contain processing rules.

2617 The overall size and scope of changes to the specification set documents will informally dictate whether a
2618 set of changes constitutes a major or minor revision. In general, if the specification set is backwards
2619 compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain
2620 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
2621 revision.

2622 4.1.1 Schema Version

2623 As a non-normative documentation mechanism, any XML schema documents published as part of the
2624 specification set will contain a `version` attribute on the `<xs:schema>` element whose value is in the
2625 form *Major.Minor*, reflecting the specification set version in which it has been published. Validating
2626 implementations MAY use the attribute as a means of distinguishing which version of a schema is being
2627 used to validate messages, or to support multiple versions of the same logical schema.

2628 4.1.2 SAML Assertion Version

2629 The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the
2630 assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed
2631 so as to document the syntax, semantics, and processing rules of the assertions of the same version.
2632 That is, specification set version 1.0 describes assertion version 1.0, and so on.

2633 There is explicitly NO relationship between the assertion version and the target XML namespace specified
2634 for the schema definitions for that assertion version.

2635 The following processing rules apply:

- 2636 • A SAML authority MUST NOT issue any assertion with an overall *Major.Minor* assertion version
2637 number not supported by the authority.
- 2638 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
2639 supported by the relying party.
- 2640 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version
2641 number is higher than the minor assertion version number supported by the relying party. However,

2642 all assertions that share a major assertion version number MUST share the same general
2643 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For
2644 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the
2645 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the
2646 likely effects of schema evolution.)

2647 **4.1.3 SAML Protocol Version**

2648 The various SAML protocols' request and response elements contain an attribute for expressing the major
2649 and minor version of the request or response message using a string of the form *Major.Minor*. Each
2650 version of the SAML specification set will be construed so as to document the syntax, semantics, and
2651 processing rules of the protocol messages of the same version. That is, specification set version 1.0
2652 describes request and response version V1.0, and so on.

2653 There is explicitly NO relationship between the protocol version and the target XML namespace specified
2654 for the schema definitions for that protocol version.

2655 The version numbers used in SAML protocol request and response elements will match for any particular
2656 revision of the SAML specification set.

2657 **4.1.3.1 Request Version**

2658 The following processing rules apply to requests:

- 2659 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2660 SAML requester and the SAML responder.
- 2661 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD
2662 assume that the responder supports requests with the highest request version supported by the
2663 requester.
- 2664 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version
2665 number matching a response version number that the requester does not support.
- 2666 • A SAML responder MUST reject any request with a major request version number not supported by
2667 the responder.
- 2668 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2669 higher than the highest supported request version that it supports. However, all requests that share
2670 a major request version number MUST share the same general processing rules and semantics,
2671 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the
2672 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill
2673 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

2674 **4.1.4 Response Version**

2675 The following processing rules apply to responses:

- 2676 • A SAML responder MUST NOT issue a response message with a response version number higher
2677 than the request version number of the corresponding request message.
- 2678 • A SAML responder MUST NOT issue a response message with a major response version number
2679 lower than the major request version number of the corresponding request message except to
2680 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2681 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
2682 top-level `<StatusCode>` value of
2683 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting

2684 one of the following second-level values:
2685 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,
2686 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, **OR**
2687 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

2688 **4.1.5 Permissible Version Combinations**

2689 Assertions of a particular major version appear only in response messages of the same major version, as
2690 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2691 example, a V1.1 assertion **MAY** appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2692 response message, if the appropriate assertion schema is referenced during namespace importation. But
2693 a V1.0 assertion **MUST NOT** appear in a V2.0 response message because they are of different major
2694 versions.

2695 **4.2 SAML Namespace Version**

2696 XML schema documents published as part of the specification set contain one or more target
2697 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct
2698 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that
2699 part of the specification.

2700 The namespace URI references defined by the specification set will generally contain version information
2701 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI **MUST** correspond
2702 to the major and minor version of the specification set in which the namespace is first introduced and
2703 defined. This information is not typically consumed by an XML processor, which treats the namespace
2704 opaquely, but is intended to communicate the relationship between the specification set and the
2705 namespaces it defines. (This pattern is also followed by the SAML-defined URI-based identifiers that are
2706 listed in Section 8.)

2707 As a general rule, implementers can expect the namespaces (and the associated schema definitions)
2708 defined by a major revision of the specification set to remain valid and stable across minor revisions of the
2709 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but
2710 this is expected to be rare. In such cases, the older namespaces and their associated definitions should
2711 be expected to remain valid until a major specification set revision.

2712 **4.2.1 Schema Evolution**

2713 In general, maintaining namespace stability while adding or changing the content of a schema are
2714 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2715 older implementations will react to any given change, making forward compatibility difficult to achieve.
2716 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
2717 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),
2718 implementations should expect forward-compatible schema changes in minor revisions, allowing new
2719 messages to validate against older schemas.

2720 Implementations **SHOULD** expect and be prepared to deal with new extensions and message types in
2721 accordance with the processing rules laid out for those types. Minor revisions **MAY** introduce new types
2722 that leverage the extension facilities described in Section 7. Older implementations **SHOULD** reject such
2723 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
2724 include new query, statement, or condition types.

2725

5 SAML and XML Signature Syntax and Processing

2726 SAML assertions and SAML protocol request and response messages may be signed, with the following
2727 benefits. An assertion signed by the SAML authority supports assertion integrity, authentication of the
2728 SAML authority to a SAML relying party, and, if the signature is based on the SAML authority's public-
2729 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the
2730 message originator supports message integrity, authentication of message origin to a destination, and, if
2731 the signature is based on the originator's public-private key pair, non-repudiation of origin.

2732 A digital signature is not always required in SAML. For example, in some circumstances, signatures may
2733 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing
2734 protocol response message. "Inherited" signatures should be used with care when the contained object
2735 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context
2736 must be retained to allow validation, exposing the XML content and adding potentially unnecessary
2737 overhead. As another example, the SAML relying party or SAML requester may have obtained an
2738 assertion or protocol message from the SAML authority or SAML responder directly (with no
2739 intermediaries) through a secure channel, with the SAML authority or SAML responder having
2740 authenticated to the relying party or SAML responder by some means other than a digital signature.

2741 Many different techniques are available for "direct" authentication and secure channel establishment
2742 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, and so on. In
2743 addition, the applicable security requirements depend on the communicating applications and the nature
2744 of the assertion or message transported. It is RECOMMENDED that, in all other contexts, digital
2745 signatures be used for assertions and request and response messages. Specifically:

- 2746 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority
2747 SHOULD be signed by the SAML authority.
- 2748 • A SAML protocol message arriving at a destination from an entity other than the originating sender
2749 SHOULD be signed by the sender.

2750 Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that
2751 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are
2752 intended to be the primary SAML signature mechanism, but this specification attempts to ensure
2753 compatibility with profiles that may require other mechanisms.

2754 Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be
2755 enveloped.

2756 5.1 Signing Assertions

2757 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as
2758 described in Section 2.

2759 5.2 Request/Response Signing

2760 All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected
2761 in the schema as described in Section 3.

2762 5.3 Signature Inheritance

2763 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
2764 or a request or response, which may be signed. When a SAML assertion does not contain a
2765 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
2766 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,

2767 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
2768 interpretation should be equivalent to the case where the assertion itself was signed with the same key
2769 and signature options.

2770 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
2771 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY
2772 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
2773 information from the surrounding context, but no such inheritance should be inferred unless specifically
2774 identified by the profile.

2775 **5.4 XML Signature Profile**

2776 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
2777 and many choices. This section details constraints on these facilities so that SAML processors do not
2778 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
2779 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**
2780 attribute on `<Assertion>` and the various request and response elements. These attributes are
2781 collectively referred to in this section as the identifier attributes.

2782 **5.4.1 Signing Formats and Algorithms**

2783 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
2784 detached.

2785 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
2786 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
2787 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

2788 **5.4.2 References**

2789 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the
2790 enclosing root element. The assertion's or protocol message's root element may or may not be the root
2791 element of the actual XML document containing the signed assertion or protocol message.

2792 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
2793 value of the root element of the message being signed. For example, if the attribute value is "foo", then
2794 the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

2795 **5.4.3 Canonicalization Method**

2796 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
2797 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a
2798 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over
2799 SAML messages embedded in an XML context can be verified independent of that context.

2800 **5.4.4 Transforms**

2801 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
2802 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
2803 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
2804 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2805 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
2806 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can

2807 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
2808 applying the transforms manually to the content and reverifying the result as consisting of the same SAML
2809 message.

2810 5.4.5 KeyInfo

2811 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of
2812 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be
2813 absent.

2814 5.4.6 Binding Between Statements in a Multi-Statement Assertion

2815 Use of signing does not affect the semantics of statements within assertions in any way, as stated in
2816 Section 2.

2817 5.4.7 Example

2818 Following is an example of a signed response containing a signed assertion. Line breaks have been
2819 added for readability; the signatures are not valid and cannot be successfully verified.

```
2820 <Response  
2821   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"  
2822   ID="_c7055387-af61-4fce-8b98-e2927324b306"  
2823   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"  
2824   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
2825   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">  
2826 <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>  
2827 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
2828 <ds:SignedInfo>  
2829 <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
2830 c14n#" />  
2831 <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
2832 <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">  
2833 <ds:Transforms>  
2834 <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-  
2835 signature" />  
2836 <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">  
2837   <InclusiveNamespaces PrefixList="#default saml samlp ds xs xsi"  
2838     xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2839 </ds:Transform>  
2840 </ds:Transforms>  
2841 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
2842 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>  
2843 </ds:Reference>  
2844 </ds:SignedInfo>  
2845 <ds:SignatureValue>  
2846 x/GyPbzmfEe85pGD3c1aXG4VspB9V9jGcJwCRKrtwPS6vdVNCcY5rHaFPYwKf+5  
2847 ELYcPzx+pXlh43SmwviCgXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D  
2848 w6vKhaqledl0BYyrIzb4KkH04ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>  
2849 <ds:KeyInfo>  
2850 <ds:X509Data>  
2851 <ds:X509Certificate>  
2852 MIIcYjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT  
2853 MRIwEAYDVQQQEiEwLXhXNjB25zaW4xEDAQBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT  
2854 FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2l2b2VzZiBJ  
2855 bmZvcmlhdG1vbiBUZWNobm9sb2d5MSUwIiwYDVQDExxIRVBLSSBTZXXJ2ZXIgcQ0Eg  
2856 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowYsYx  
2857 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbWVjESMBAGA1UEBxMjQW5uIEFy  
2858 Ym9yMQ4wDAYDVQQKEwVlV2Q0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJlZmVjZmVj  
2859 dTEEnMCUGCSqGSIB3DQEJARYYcm9vdEBzaGlms5pbnRlcm5ldIuZWR1MIGfMA0G  
2860 CSqGSIB3DQEBAQUAA4GNADCBiQKBgQDZSAB2sxvhAXnXVIVT8vuRay+x50z7GJj
```

2861 IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIAoAPSZBl13R6+KYIE7x4XAWIrcP+
2862 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
2863 pmqOIFGTWQIDAQABox0wGzAMBGNVHRMBaf8EAjAAMAsGA1UdDwQEAWIFoDANBgkq
2864 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2n
2865 qgi7lFV6MDkhmTvtTqBtmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
2866 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpRllylGPdiowMNTreG8cCx3w/w==
2867 </ds:X509Certificate>
2868 </ds:X509Data>
2869 </ds:KeyInfo>
2870 </ds:Signature>
2871 <Status><StatusCode
2872 Value="urn:oasis:names:tc:SAML:2.0:status:Success"/></Status>
2873 <Assertion
2874 ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2875 IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
2876 xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
2877 <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
2878 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2879 <ds:SignedInfo>
2880 <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
2881 c14n#" />
2882 <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2883 <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
2884 <ds:Transforms>
2885 <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
2886 signature" />
2887 <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2888 <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
2889 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2890 </ds:Transform>
2891 </ds:Transforms>
2892 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2893 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
2894 </ds:Reference>
2895 </ds:SignedInfo>
2896 <ds:SignatureValue>
2897 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgzpkJN9CMLG8ENR4Nrw+n
2898 7iyzixBvKX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOT6qaAsNdeCyG86jmt3TD
2899 MWuL/cBUj2OtBZQMFn7jQ9YB7klIz3RqVL+wNmeWI4=</ds:SignatureValue>
2900 <ds:KeyInfo>
2901 <ds:X509Data>
2902 <ds:X509Certificate>
2903 MIIcYjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgakkCzAJBgNVBAYTA1VT
2904 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
2905 FlVuaXZlcnNpdHkkgb2YgV2l2Y29uc2luMSswKQYDVQQLJEaXZpc2lvbiBvZiBJ
2906 bmZvcmlhdG1vbiBUZWNobm9sb2d5MSUwIiwYDVQDExxIRVBLSSBTXZJ2ZXIgzQ0Eg
2907 LS0gMjAAMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVoWovYsX
2908 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAQA1UEBxMjQW5uIEFY
2909 Ym9yMQ4wDAYDVQKKEwVUVVQ0FJRDEcMBoGA1UEAxMTC2hpYjEuaW50ZXJuzXQyLmVl
2910 dTENMCUGCSqGSIb3DQEJARYYcm9vdEBzAgLiMS5pbmRlcm5ldDIuZWZWR1MIGfMAOG
2911 CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZsAb2sXvAhAXnXVIvtX8vuRay+x50z7GJj
2912 IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIAoAPSZBl13R6+KYIE7x4XAWIrcP+
2913 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
2914 pmqOIFGTWQIDAQABox0wGzAMBGNVHRMBaf8EAjAAMAsGA1UdDwQEAWIFoDANBgkq
2915 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2n
2916 qgi7lFV6MDkhmTvtTqBtmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
2917 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpRllylGPdiowMNTreG8cCx3w/w==
2918 </ds:X509Certificate>
2919 </ds:X509Data>
2920 </ds:KeyInfo>
2921 </ds:Signature>
2922 <Subject>
2923 <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
2924 scott@example.org
2925 </NameID>
2926 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer" />

```
2927 </Subject>
2928 <Conditions NotBefore="2003-04-17T00:46:02Z" NotOnOrAfter="2003-04-
2929 17T00:51:02Z">
2930   <AudienceRestriction>
2931     <Audience>http://www.opensaml.org/SP</Audience>
2932   </AudienceRestriction>
2933 </Conditions>
2934 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
2935   <AuthnContext>
2936     <AuthnContextClassRef>
2937       urn:oasis:names:tc:SAML:2.0:ac:classes:Password
2938     </AuthnContextClassRef>
2939   </AuthnContext>
2940 </AuthnStatement>
2941 </Assertion>
2942 </Response>
```

2943

6 SAML and XML Encryption Syntax and Processing

2944
2945
2946
2947

Encryption is used as the means to implement confidentiality. The most common motives for confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for competitive advantage or similar reasons. Confidentiality may also be required to insure the effectiveness of some other security mechanism. For example, a secret password or key may be encrypted.

2948

Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

2949
2950
2951

- Communications confidentiality may be provided by mechanisms associated with a particular binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS or SOAP Message Security mechanisms for confidentiality.

2952
2953
2954

- A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>` element within `<SubjectConfirmationData>`, which permits keys or other secrets to be encrypted.

2955

- An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.

2956

- The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.3.

2957

- An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

2958

6.1 General Considerations

2959
2960
2961
2962
2963

Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace the cleartext information in the same location within the XML instance. The `<EncryptedData>` element's `Type` attribute SHOULD be used and, if it is present, MUST have the value `http://www.w3.org/2001/04/xmleenc#Element`.

2964
2965

Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

2966

6.2 Combining Signatures and Encryption

2967
2968
2969

Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in the reverse order that signing and encryption were performed.

2970
2971

- When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and in place, and then the element encrypted.

2972
2973
2974

- When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be performed first and then the signature calculated over the assertion or message containing the encrypted element.

2975

7 SAML Extensibility

2976 SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas.
2977 An example of an application that extends SAML assertions is the Liberty Protocols and Schema
2978 Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions
2979 and protocols.

2980 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles of use,
2981 which can be combined with extensions to put the SAML framework to new uses.

7.1 Schema Extension

2983 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
2984 elements can serve as the head element of a substitution group. However, SAML types are not defined as
2985 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only
2986 elements and types that have been specifically designed to support extensibility.

7.1.1 Assertion Schema Extension

2988 The SAML assertion schema is designed to permit separate processing of the assertion package and the
2989 statements it contains, if the extension mechanism is used for either part.

2990 The following elements are intended specifically for use as extension points in an extension schema; their
2991 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 2992 • `<BaseID>` and **BaseIDAbstractType**
- 2993 • `<Condition>` and **ConditionAbstractType**
- 2994 • `<Statement>` and **StatementAbstractType**

2995 The following constructs that are directly usable as part of SAML are particularly interesting targets for
2996 extension:

- 2997 • `<AuthnStatement>` and **AuthnStatementType**
- 2998 • `<AttributeStatement>` and **AttributeStatementType**
- 2999 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
- 3000 • `<AudienceRestriction>` and **AudienceRestrictionType**
- 3001 • `<ProxyRestriction>` and **ProxyRestrictionType**
- 3002 • `<OneTimeUse>` and **OneTimeUseType**

7.1.2 Protocol Schema Extension

3004 The following SAML protocol elements are intended specifically for use as extension points in an
3005 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
3006 type:

- 3007 • `<Request>` and **RequestAbstractType**
- 3008 • `<SubjectQuery>` and **SubjectQueryAbstractType**

3009 The following constructs that are directly usable as part of SAML are particularly interesting targets for
3010 extension:

- 3011 • <AuthnQuery> and **AuthnQueryType**
- 3012 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**
- 3013 • <AttributeQuery> and **AttributeQueryType**
- 3014 • **StatusResponseType**

3015 7.2 Schema Wildcard Extension Points

3016 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes
3017 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension
3018 schema.

3019 7.2.1 Assertion Extension Points

3020 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- 3021 • <SubjectConfirmationData>: Uses **xs:anyType**, which allows any sub-elements and
3022 attributes.
- 3023 • <AuthnContextDecl>: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3024 • <AttributeValue>: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3025 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other
3026 namespaces with lax schema validation processing.

3027 The following constructs in the assertion schema allow arbitrary global attributes:

- 3028 • <Attribute> and **AttributeType**

3029 7.2.2 Protocol Extension Points

3030 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- 3031 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema
3032 validation processing.
- 3033 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax
3034 schema validation processing.
- 3035 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with
3036 lax schema validation processing. (It is specifically intended to carry a SAML request or response
3037 message element, however.)

3038 7.3 Identifier Extension

3039 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
3040 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
3041 However, it is always possible to define additional URI-based identifiers for these purposes. It is
3042 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should
3043 the meaning of a given URI used as such an identifier significantly change, or be used to mean two
3044 different things.

3045 8 SAML-Defined Identifiers

3046 The following sections define URI-based identifiers for common resource access actions, subject name
3047 identifier formats, and attribute name formats.

3048 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of
3049 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
3050 have one of the following stems, according to the specification set version in which they were first
3051 introduced:

```
3052 urn:oasis:names:tc:SAML:1.0:  
3053 urn:oasis:names:tc:SAML:1.1:  
3054 urn:oasis:names:tc:SAML:2.0:
```

3055 8.1 Action Namespace Identifiers

3056 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to
3057 common sets of actions to perform on resources.

3058 8.1.1 Read/Write/Execute/Delete/Control

3059 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

3060 Defined actions:

3061 `Read Write Execute Delete Control`

3062 These actions are interpreted as follows:

3063 `Read`

3064 The subject may read the resource.

3065 `Write`

3066 The subject may modify the resource.

3067 `Execute`

3068 The subject may execute the resource.

3069 `Delete`

3070 The subject may delete the resource.

3071 `Control`

3072 The subject may specify the access control policy for the resource.

3073 8.1.2 Read/Write/Execute/Delete/Control with Negation

3074 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

3075 Defined actions:

3076 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

3077 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions
3078 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
3079 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is
3080 affirmatively denied read permission.

3081 A SAML authority MUST NOT authorize both an action and its negated form.

3082 **8.1.3 Get/Head/Put/Post**

3083 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3084 Defined actions:

3085 GET HEAD PUT POST

3086 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
3087 the GET action on a resource is authorized to retrieve it.

3088 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST
3089 actions to the write permission. The correspondence is not exact however since an HTTP GET operation
3090 may cause data to be modified and a POST operation may cause modification to a resource other than
3091 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

3092 **8.1.4 UNIX File Permissions**

3093 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3094 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3095 The action string is a four-digit numeric code:

3096 *extended user group world*

3097 Where the *extended* access permission has the value

3098 +2 if sgid is set

3099 +4 if suid is set

3100 The *user group* and *world* access permissions have the value

3101 +1 if execute permission is granted

3102 +2 if write permission is granted

3103 +4 if read permission is granted

3104 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
3105 and execute; and world read.

3106 **8.2 Attribute Name Format Identifiers**

3107 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex
3108 type to refer to the classification of the attribute name for purposes of interpreting the name.

3109 **8.2.1 Unspecified**

3110 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3111 The interpretation of the attribute name is left to individual implementations.

3112 8.2.2 URI Reference

3113 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

3114 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML
3115 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-
3116 specific. See [SAMLProf] for attribute profiles that make use of this identifier.

3117 8.2.3 Basic

3118 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

3119 The class of strings acceptable as the attribute name **MUST** be drawn from the set of values belonging to
3120 the primitive type **xs:Name** as defined in [Schema2] §3.3.6 . See [SAMLProf] for attribute profiles that
3121 make use of this identifier.

3122 8.3 Name Identifier Format Identifiers

3123 The following identifiers **MAY** be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
3124 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the
3125 associated processing rules, if any.

3126 **Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for
3127 SAML V2.0.

3128 8.3.1 Unspecified

3129 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3130 The interpretation of the content of the element is left to individual implementations.

3131 8.3.2 Email Address

3132 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3133 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
3134 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that
3135 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in
3136 parentheses) after it, and is not surrounded by "<" and ">".

3137 8.3.3 X.509 Subject Name

3138 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3139 Indicates that the content of the element is in the form specified for the contents of the
3140 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
3141 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
3142 differ from the rules given in IETF RFC 2253 [RFC 2253].

3143 8.3.4 Windows Domain Qualified Name

3144 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3145 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3146 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator
3147 MAY be omitted.

3148 **8.3.5 Kerberos Principal Name**

3149 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3150 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3151 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
3152 realm are described in [RFC 1510].

3153 **8.3.6 Entity Identifier**

3154 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3155 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3156 (such as a SAML authority) or is a participant in SAML profiles (such as a service provider supporting the
3157 browser SSO profile). Such an identifier can be used in the <Issuer> element to identify the issuer of a
3158 SAML request, response, or assertion, or within the <NameID> element to make assertions about system
3159 entities that can issue SAML requests, responses, and assertions. It can also be used in other elements
3160 and attributes whose purpose is to identify a system entity in various protocol exchanges.

3161 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3162 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3163 **8.3.7 Persistent Identifier**

3164 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3165 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3166 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
3167 generated by identity providers MUST be constructed using pseudo-random values that have no
3168 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
3169 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
3170 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3171 The element's *NameQualifier* attribute, if present, MUST contain the unique identifier of the identity
3172 provider that generated the identifier (see section 8.3.6). It MAY be omitted if the value can be derived
3173 from the context of the message containing the element, such as the issuer of a protocol message or an
3174 assertion containing the identifier in its subject. Note that a different system entity might later issue its own
3175 protocol message or assertion containing the identifier; the *NameQualifier* does not change in this case,
3176 but MUST continue to identify the entity that originally created the identifier (and MUST NOT be omitted in
3177 such a case).

3178 The element's *SPNameQualifier* attribute, if present, MUST contain the unique identifier of the service
3179 provider or affiliation of providers for whom the identifier was generated (see section 8.3.6). It MAY be
3180 omitted if the element is contained in a message intended only for consumption directly by the service
3181 provider, and the value would be the name of that service provider.

3182 The element's *SPProvidedID* attribute MUST contain the alternative identifier of the principal most
3183 recently set by the service provider or affiliation, if any (see section 3.6). If no such identifier has been
3184 established, then the attribute MUST be omitted.

3185 Persistent identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear text
3186 with providers other than the providers that have established the shared identifier. Furthermore, they
3187 MUST NOT appear in log files or similar locations without appropriate controls and protections.

3188 Deployments without such requirements are free to use other kinds of identifiers in their SAML
3189 exchanges, but MUST NOT overload this format with persistent but non-opaque values

3190 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3191 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3192 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3193 different and does not affect the behavior of the identity provider or any processing rules specific to
3194 persistent identifiers in the protocols defined in this specification.

3195 **8.3.8 Transient Identifier**

3196 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

3197 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3198 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3199 accordance with the rules for SAML identifiers (see Section 1.2.4), and MUST NOT exceed a length of
3200 256 characters.

3201 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3202 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
3203 accordance with the rules specified in Section 8.3.7.

3204 **8.4 Consent Identifiers**

3205 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType**
3206 complex type to communicate whether a user gave consent, and under what conditions, for the request.

3207 **8.4.1 Unspecified**

3208 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unspecified

3209 No claim as to user consent is being made.

3210 **8.4.2 Obtained**

3211 **URI:** urn:oasis:names:tc:SAML:2.0:consent:obtained

3212 Indicates that a user's consent has been obtained by the issuer of the request.

3213 **8.4.3 Prior**

3214 **URI:** urn:oasis:names:tc:SAML:2.0:consent:prior

3215 Indicates that a user's consent has been obtained by the issuer of the request at some point prior to the
3216 action that initiated the request.

3217 **8.4.4 Implicit**

3218 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3219 Indicates that a user's consent has been implicitly obtained by the issuer of the request during the action
3220 that initiated the request, as part of a broader indication of consent. Implicit consent is typically more
3221 proximal to the action in time and presentation than prior consent, such as part of a session of activities.

3222 **8.4.5 Explicit**

3223 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3224 Indicates that a user's consent has been explicitly obtained by the issuer of the request during the action
3225 that initiated the request.

3226 **8.4.6 Unavailable**

3227 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3228 Indicates that the issuer of the request did not obtain consent.

3229 **8.4.7 Inapplicable**

3230 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3231 Indicates that the issuer of the request does not believe that they need to obtain or report consent.

3232

9 References

3233 The following works are cited in the body of this specification.

3234

9.1 Normative References

- 3235 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3236
- 3237 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- 3238
- 3239
- 3240 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3241
- 3242 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3243
- 3244 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web Consortium. Note that this specification normatively references [XMLEnc-XSD], listed below.
- 3245
- 3246
- 3247
- 3248 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium. <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3249
- 3250 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January 1999. <http://www.w3.org/TR/REC-xml-names>.
- 3251
- 3252 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this specification normatively references [XMLSig-XSD], listed below.
- 3253
- 3254
- 3255 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium. <http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd>.
- 3256
- 3257

3258

9.2 Non-Normative References

- 3259 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty Alliance Project, January 2003, http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf.
- 3260
- 3261
- 3262
- 3263 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December 1978.
- 3264
- 3265
- 3266 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*. IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 3267
- 3268 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF RFC 2459, January 1999. <http://www.ietf.org/rfc/rfc2459.txt>.
- 3269
- 3270
- 3271 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3272
- 3273 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- 3274

3275 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
3276 <http://www.ietf.org/rfc/rfc2246.txt>.

3277 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*
3278 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.
3279 <http://www.ietf.org/rfc/rfc2253.txt>.

3280 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF
3281 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3282 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.
3283 <http://www.ietf.org/rfc/rfc2630.txt>.

3284 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.
3285 <http://www.ietf.org/rfc/rfc2822.txt>.

3286 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,
3287 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3288 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF
3289 RFC 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3290 [SAMLAuthnCxt] J. Kemp et al., *Authentication Context for the OASIS Security Assertion Markup*
3291 *Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-
3292 saml-authn-context-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
3293 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3294 [SAMLBind] S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*
3295 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-
3296 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.

3297 [SAMLMeta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language*
3298 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-metadata-
3299 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.

3300 [SAMLProf] S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*
3301 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-profiles-
3302 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.

3303 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion*
3304 *Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID
3305 sstc-saml-conformance-2.0-cd-02. [http://www.oasis-](http://www.oasis-open.org/committees/security/)
3306 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3307 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*
3308 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf)
3309 [open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3310 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language*
3311 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-glossary-
3312 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.

3313 [SAMLPSchema] S. Cantor et al., *SAML protocols schema*. OASIS SSTC, September 2004.
3314 Document ID sstc-saml-schema-protocol-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
3315 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3316 [SAMLSecure] F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*
3317 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004.
3318 Document ID sstc-saml-sec-consider-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
3319 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3320 [SAMLXSD] S. Cantor et al., *SAML assertions schema*. OASIS SSTC, September 2004.
3321 Document ID sstc-saml-schema-assertion-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
3322 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

- 3323 **[SAML-TechOvw]** J. Hughes et al. SAML Technical Overview. OASIS, July 2004. Document ID
3324 oasis-sstc-saml-tech-overview-2.0. [http://www.oasis-
open.org/committees/security/](http://www.oasis-
3325 open.org/committees/security/).
- 3326 **[UNICODE-C]** M. Davis, M. J. Dürst. *Unicode Normalization Forms*. UNICODE Consortium,
3327 March 2001. <http://www.unicode.org/unicode/reports/tr15/tr15-21.html>.
- 3328 **[W3C-CHAR]** M. J. Dürst. *Requirements for String Identity Matching and String Indexing*. World
3329 Wide Web Consortium, July 1998. <http://www.w3.org/TR/WD-charreq>.
- 3330 **[W3C-CharMod]** M. J. Dürst. *Character Model for the World Wide Web 1.0: Normalization*. World
3331 Wide Web Consortium, February 2004. <http://www.w3.org/TR/charmod-norm/>.
- 3332 **[X.500]** ITU-T Recommendation X.501: Information Technology - Open Systems
3333 Interconnection - The Directory: Models. 1993.
- 3334 **[XACML]** eXtensible Access Control Markup Language (XACML), product of the OASIS
3335 XACML TC. See <http://www.oasis-open.org/committees/xacml>.
- 3336 **[XML-ID]** J. Marsh et al., *xml:id Version 1.0*, W3C, April 2004. [http://www.w3.org/TR/xml-
id/](http://www.w3.org/TR/xml-
3337 id/).

3338

Appendix A. Acknowledgments

3339
3340

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379

- Conor Cahill, AOL
- John Hughes, ATOS Origin
- Hal Lockhart, BEA Systems
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Tim Alsop, CyberSafe Limited
- Paul Madsen, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Paula Austel, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Prateek Mishra, Netegrity
- Forest Yin, Netegrity
- Peter Davis, Neustar
- Frederick Hirsch, Nokia
- John Kemp, Nokia
- Senthil Sengodan, Nokia
- Scott Kiestler, Novell
- Cameron Morris, Novell
- Charles Knouse, Oblix
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Jeff Hodges, Sun Microsystems
- Eve Maler, Sun Microsystems

- 3380 • Ronald Monzillo, Sun Microsystems
- 3381 • Emily Xu, Sun Microsystems
- 3382 • Mike Beach, Boeing
- 3383 • Greg Whitehead, Trustgenix
- 3384 •

3385 The editors also would like to acknowledge the following people for their contributions to previous versions
3386 of the OASIS Security Assertions Markup Language Standard:

- 3387 • Stephen Farrell, Baltimore Technologies
- 3388 • David Orchard, BEA Systems
- 3389 • Krishna Sankar, Cisco Systems
- 3390 • Zahid Ahmed, CommerceOne
- 3391 • Carlisle Adams, Entrust
- 3392 • Tim Moses, Entrust
- 3393 • Nigel Edwards, Hewlett-Packard
- 3394 • Joe Pato, Hewlett-Packard
- 3395 • Bob Blakley, IBM
- 3396 • Marlena Erdos, IBM
- 3397 • Marc Chanliau, Netegrity
- 3398 • Chris McLaren, Netegrity
- 3399 • Lynne Rosenthal, NIST
- 3400 • Mark Skall, NIST
- 3401 • Simon Godik, Overxeer
- 3402 • Charles Norwood, SAIC
- 3403 • Evan Prodromou, Securant
- 3404 • Robert Griffin, RSA Security (former editor)
- 3405 • Sai Allarvarpu, Sun Microsystems
- 3406 • Chris Ferris, Sun Microsystems
- 3407 • Emily Xu, Sun Microsystems
- 3408 • Mike Myers, Traceroute Security
- 3409 • Phillip Hallam-Baker, VeriSign (former editor)
- 3410 • James Vanderbeek, Vodafone
- 3411 • Mark O'Neill, Vordel
- 3412 • Tony Palmer, Vordel

3413

3414 Finally, the editors wish to acknowledge the following people for their contributions of material used as
3415 input to the OASIS Security Assertions Markup Language specifications:

- 3416 • Thomas Gross, IBM
- 3417 • Birgit Pfitzmann, IBM

3418

Appendix B. Notices

3419 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3420 might be claimed to pertain to the implementation or use of the technology described in this document or
3421 the extent to which any license under such rights might or might not be available; neither does it represent
3422 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
3423 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
3424 available for publication and any assurances of licenses to be made available, or the result of an attempt
3425 made to obtain a general license or permission for the use of such proprietary rights by implementors or
3426 users of this specification, can be obtained from the OASIS Executive Director.

3427 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
3428 other proprietary rights which may cover technology that may be required to implement this specification.
3429 Please address the information to the OASIS Executive Director.

3430 **Copyright © OASIS Open 2004. All Rights Reserved.**

3431 This document and translations of it may be copied and furnished to others, and derivative works that
3432 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
3433 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
3434 this paragraph are included on all such copies and derivative works. However, this document itself may
3435 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
3436 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
3437 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
3438 into languages other than English.

3439 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3440 or assigns.

3441 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3442 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3443 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3444 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.