



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

OASIS Standard, 15 March 2005

Document identifier:

saml-core-2.0-os

Location:

<http://docs.oasis-open.org/security/saml/v2.0/>

Editors:

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

SAML V2.0 Contributors:

Conor P. Cahill, AOL
John Hughes, Atos Origin
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rebekah Metz, Booz Allen Hamilton
Rick Randall, Booz Allen Hamilton
Thomas Wisniewski, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Nick Ragouzis, Individual
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Peter C Davis, Neustar
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Paul Madsen, NTT
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems

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

45 **Abstract:**

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

48 **Status:**

49 This is an **OASIS Standard** document produced by the Security Services Technical Committee. It
50 was approved by the OASIS membership on 1 March 2005.

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

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

Table of Contents

61	1 Introduction.....	7
62	1.1 Notation.....	7
63	1.2 Schema Organization and Namespaces.....	8
64	1.3 Common Data Types.....	8
65	1.3.1 String Values.....	8
66	1.3.2 URI Values.....	9
67	1.3.3 Time Values.....	9
68	1.3.4 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 Complex Type NameIDType.....	13
74	2.2.3 Element <NameID>.....	14
75	2.2.4 Element <EncryptedID>.....	14
76	2.2.5 Element <Issuer>.....	15
77	2.3 Assertions.....	15
78	2.3.1 Element <AssertionIDRef>.....	15
79	2.3.2 Element <AssertionURIRef>.....	15
80	2.3.3 Element <Assertion>.....	15
81	2.3.4 Element <EncryptedAssertion>.....	17
82	2.4 Subjects.....	17
83	2.4.1 Element <Subject>.....	18
84	2.4.1.1 Element <SubjectConfirmation>.....	18
85	2.4.1.2 Element <SubjectConfirmationData>.....	19
86	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	20
87	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
88	2.5 Conditions.....	21
89	2.5.1 Element <Conditions>.....	21
90	2.5.1.1 General Processing Rules.....	22
91	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
92	2.5.1.3 Element <Condition>.....	23
93	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	23
94	2.5.1.5 Element <OneTimeUse>.....	24
95	2.5.1.6 Element <ProxyRestriction>.....	25
96	2.6 Advice.....	25
97	2.6.1 Element <Advice>.....	26
98	2.7 Statements.....	26
99	2.7.1 Element <Statement>.....	26
100	2.7.2 Element <AuthnStatement>.....	26
101	2.7.2.1 Element <SubjectLocality>.....	28
102	2.7.2.2 Element <AuthnContext>.....	28
103	2.7.3 Element <AttributeStatement>.....	29
104	2.7.3.1 Element <Attribute>.....	29
105	2.7.3.1.1 Element <AttributeValue>.....	30
106	2.7.3.2 Element <EncryptedAttribute>.....	31
107	2.7.4 Element <AuthzDecisionStatement>.....	31
108	2.7.4.1 Simple Type DecisionType.....	33
109	2.7.4.2 Element <Action>.....	33

110	2.7.4.3 Element <Evidence>.....	34
111	3 SAML Protocols.....	35
112	3.1 Schema Header and Namespace Declarations.....	35
113	3.2 Requests and Responses.....	36
114	3.2.1 Complex Type RequestAbstractType.....	36
115	3.2.2 Complex Type StatusResponseType.....	38
116	3.2.2.1 Element <Status>.....	39
117	3.2.2.2 Element <StatusCode>.....	39
118	3.2.2.3 Element <StatusMessage>.....	42
119	3.2.2.4 Element <StatusDetail>.....	42
120	3.3 Assertion Query and Request Protocol.....	42
121	3.3.1 Element <AssertionIDRequest>.....	42
122	3.3.2 Queries.....	42
123	3.3.2.1 Element <SubjectQuery>.....	42
124	3.3.2.2 Element <AuthnQuery>.....	43
125	3.3.2.2.1 Element <RequestedAuthnContext>.....	44
126	3.3.2.3 Element <AttributeQuery>.....	45
127	3.3.2.4 Element <AuthzDecisionQuery>.....	46
128	3.3.3 Element <Response>.....	46
129	3.3.4 Processing Rules.....	47
130	3.4 Authentication Request Protocol.....	47
131	3.4.1 Element <AuthnRequest>.....	48
132	3.4.1.1 Element <NameIDPolicy>.....	50
133	3.4.1.2 Element <Scoping>.....	51
134	3.4.1.3 Element <IDPList>.....	52
135	3.4.1.3.1 Element <IDPEntry>.....	52
136	3.4.1.4 Processing Rules.....	53
137	3.4.1.5 Proxying.....	54
138	3.4.1.5.1 Proxying Processing Rules.....	54
139	3.5 Artifact Resolution Protocol.....	55
140	3.5.1 Element <ArtifactResolve>.....	56
141	3.5.2 Element <ArtifactResponse>.....	56
142	3.5.3 Processing Rules.....	56
143	3.6 Name Identifier Management Protocol.....	57
144	3.6.1 Element <ManageNameIDRequest>.....	57
145	3.6.2 Element <ManageNameIDResponse>.....	58
146	3.6.3 Processing Rules.....	58
147	3.7 Single Logout Protocol.....	59
148	3.7.1 Element <LogoutRequest>.....	60
149	3.7.2 Element <LogoutResponse>.....	60
150	3.7.3 Processing Rules.....	61
151	3.7.3.1 Session Participant Rules.....	61
152	3.7.3.2 Session Authority Rules.....	62
153	3.8 Name Identifier Mapping Protocol.....	63
154	3.8.1 Element <NameIDMappingRequest>.....	63
155	3.8.2 Element <NameIDMappingResponse>.....	64
156	3.8.3 Processing Rules.....	64
157	4 SAML Versioning.....	65
158	4.1 SAML Specification Set Version.....	65
159	4.1.1 Schema Version.....	65
160	4.1.2 SAML Assertion Version.....	65
161	4.1.3 SAML Protocol Version.....	66
162	4.1.3.1 Request Version.....	66

163	4.1.3.2 Response Version.....	66
164	4.1.3.3 Permissible Version Combinations.....	67
165	4.2 SAML Namespace Version.....	67
166	4.2.1 Schema Evolution.....	67
167	5 SAML and XML Signature Syntax and Processing.....	68
168	5.1 Signing Assertions.....	68
169	5.2 Request/Response Signing.....	68
170	5.3 Signature Inheritance.....	68
171	5.4 XML Signature Profile.....	69
172	5.4.1 Signing Formats and Algorithms.....	69
173	5.4.2 References.....	69
174	5.4.3 Canonicalization Method.....	69
175	5.4.4 Transforms.....	69
176	5.4.5 KeyInfo.....	70
177	5.4.6 Example.....	70
178	6 SAML and XML Encryption Syntax and Processing.....	73
179	6.1 General Considerations.....	73
180	6.2 Combining Signatures and Encryption.....	73
181	7 SAML Extensibility.....	74
182	7.1 Schema Extension.....	74
183	7.1.1 Assertion Schema Extension.....	74
184	7.1.2 Protocol Schema Extension.....	74
185	7.2 Schema Wildcard Extension Points.....	75
186	7.2.1 Assertion Extension Points.....	75
187	7.2.2 Protocol Extension Points.....	75
188	7.3 Identifier Extension.....	75
189	8 SAML-Defined Identifiers.....	76
190	8.1 Action Namespace Identifiers.....	76
191	8.1.1 Read/Write/Execute/Delete/Control.....	76
192	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	76
193	8.1.3 Get/Head/Put/Post.....	77
194	8.1.4 UNIX File Permissions.....	77
195	8.2 Attribute Name Format Identifiers.....	77
196	8.2.1 Unspecified.....	77
197	8.2.2 URI Reference.....	78
198	8.2.3 Basic.....	78
199	8.3 Name Identifier Format Identifiers.....	78
200	8.3.1 Unspecified.....	78
201	8.3.2 Email Address.....	78
202	8.3.3 X.509 Subject Name.....	78
203	8.3.4 Windows Domain Qualified Name.....	78
204	8.3.5 Kerberos Principal Name.....	79
205	8.3.6 Entity Identifier.....	79
206	8.3.7 Persistent Identifier.....	79
207	8.3.8 Transient Identifier.....	80
208	8.4 Consent Identifiers.....	80
209	8.4.1 Unspecified.....	80
210	8.4.2 Obtained.....	80
211	8.4.3 Prior.....	80
212	8.4.4 Implicit.....	81
213	8.4.5 Explicit.....	81

214	8.4.6 Unavailable.....	81
215	8.4.7 Inapplicable.....	81
216	9 References.....	82
217	9.1 Normative References.....	82
218	9.2 Non-Normative References.....	82
219	Appendix A. Acknowledgments.....	84
220	Appendix B. Notices.....	86
221		

222 1 Introduction

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

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

234 For additional explanation of SAML terms and concepts, refer to the SAML technical overview
235 [SAMLTechOvw] and the SAML glossary [SAMLGloss] . Files containing just the SAML assertion schema
236 [SAML-XSD] and protocol schema [SAMLP-XSD] are also available. The SAML conformance document
237 [SAMLConform] lists all of the specifications that comprise SAML V2.0.

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

239 1.1 Notation

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

243 Listings of SAML schemas appear like this.

244 Example code listings appear like this.

246 **Note:** Notes like this are sometimes used to highlight non-normative commentary.

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

252 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for
253 their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is
254 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 [SAMLP-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].

Prefix	XML Namespace	Comments
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].
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.

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

257 1.2 Schema Organization and Namespaces

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

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

261 The SAML request-response protocol structures are defined in a schema [SAML-PROT] associated with
262 the following XML namespace:

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

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

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

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

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

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

272 1.3 Common Data Types

273 The following sections define how to use and interpret common data types that appear throughout the
274 SAML schemas.

275 1.3.1 String Values

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

280 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that
281 have the XML Schema **xs:string** type, or a type derived from that, **MUST** be compared using an exact
282 binary comparison. In particular, SAML implementations and deployments **MUST NOT** depend on case-
283 insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific

284 formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft
285 Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

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

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

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

300 1.3.2 URI Values

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

303 Unless otherwise indicated in this specification, all URI reference values used within SAML-defined
304 elements or attributes MUST consist of at least one non-whitespace character, and are REQUIRED to be
305 absolute [RFC 2396].

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

310 1.3.3 Time Values

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

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

315 1.3.4 ID and ID Reference Values

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

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

323 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the
324 implementation. In the case that a random or pseudorandom technique is employed, the probability of two
325 randomly chosen identifiers being identical MUST be less than or equal to 2^{-128} and SHOULD be less than
326 or equal to 2^{-160} . This requirement MAY be met by encoding a randomly chosen value between 128 and

327 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom
328 generator **MUST** be seeded with unique material in order to ensure the desired uniqueness properties
329 between different systems.

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

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

2 SAML Assertions

340

341 An assertion is a package of information that supplies zero or more statements made by a **SAML**
342 **authority**; SAML authorities are sometimes referred to as **asserting parties** in discussions of assertion
343 generation and exchange, and system entities that use received assertions are known as **relying parties**.
344 (Note that these terms are different from **requester** and **responder**, which are reserved for discussions of
345 SAML protocol message exchange.)

346 SAML assertions are usually made about a **subject**, represented by the <Subject> element. However,
347 the <Subject> element is optional, and other specifications and profiles may utilize the SAML assertion
348 structure to make similar statements without specifying a subject, or possibly specifying the subject in an
349 alternate way. Typically there are a number of **service providers** that can make use of assertions about a
350 subject in order to control access and provide customized service, and accordingly they become the
351 relying parties of an asserting party called an **identity provider**.

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

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

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

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

365 The SAML technical overview [SAMLTechOvw] and glossary [SAMLGloss] provide more detailed
366 explanation of SAML terms and concepts.

2.1 Schema Header and Namespace Declarations

367

368 The following schema fragment defines the XML namespaces and other header information for the
369 assertion schema:

```
370 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
371   xmlns="http://www.w3.org/2001/XMLSchema"  
372   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
373   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
374   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  
375   elementFormDefault="unqualified"  
376   attributeFormDefault="unqualified"  
377   blockDefault="substitution"  
378   version="2.0">  
379   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
380     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-  
381 20020212/xmldsig-core-schema.xsd"/>  
382   <import namespace="http://www.w3.org/2001/04/xmlenc#"  
383     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-  
384 20021210/xenc-schema.xsd"/>  
385   <annotation>  
386     <documentation>  
387       Document identifier: saml-schema-assertion-2.0
```

```

388         Location: http://docs.oasis-open.org/security/saml/v2.0/
389         Revision history:
390         V1.0 (November, 2002):
391             Initial Standard Schema.
392         V1.1 (September, 2003):
393             Updates within the same V1.0 namespace.
394         V2.0 (March, 2005):
395             New assertion schema for SAML V2.0 namespace.
396     </documentation>
397 </annotation>
398 ...
399 </schema>

```

400 2.2 Name Identifiers

401 The following sections define the SAML constructs that contain descriptive identifiers for subjects and the
402 issuers of assertions and protocol messages.

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

410 It is possible that two or more system entities may use the same name identifier value when referring to
411 different identities. Thus, each entity may have a different understanding of that same name. SAML
412 provides **name qualifiers** to disambiguate a name identifier by effectively placing it in a federated
413 **namespace** related to the name qualifiers. SAML V2.0 allows an identifier to be qualified in terms of both
414 an asserting party and a particular relying party or affiliation, allowing identifiers to exhibit pair-wise
415 semantics, when required.

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

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

420 2.2.1 Element <BaseID>

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

424 **NameQualifier** [Optional]

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

427 **SPNameQualifier** [Optional]

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

431 The **NameQualifier** and **SPNameQualifier** attributes SHOULD be omitted unless the identifier's type
432 definition explicitly defines their use and semantics.

433 The following schema fragment defines the <BaseID> element and its **BaseIDAbstractType** complex
434 type:

```
435 <attributeGroup name="IDNameQualifiers">  
436   <attribute name="NameQualifier" type="string" use="optional"/>  
437   <attribute name="SPNameQualifier" type="string" use="optional"/>  
438 </attributeGroup>  
439 <element name="BaseID" type="saml:BaseIDAbstractType"/>  
440 <complexType name="BaseIDAbstractType" abstract="true">  
441   <attributeGroup ref="saml:IDNameQualifiers"/>  
442 </complexType>
```

443 2.2.2 Complex Type NameIDType

444 The **NameIDType** complex type is used when an element serves to represent an entity by a string-valued
445 name. It is a more restricted form of identifier than the <BaseID> element and is the type underlying both
446 the <NameID> and <Issuer> elements. In addition to the string content containing the actual identifier, it
447 provides the following optional attributes:

448 NameQualifier [Optional]

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

451 SPNameQualifier [Optional]

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

455 Format [Optional]

456 A URI reference representing the classification of string-based identifier information. See Section
457 8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute
458 and their associated descriptions and processing rules. Unless otherwise specified by an element
459 based on this type, if no `Format` value is provided, then the value
460 `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in
461 effect.

462 When a `Format` value other than one specified in Section 8.3 is used, the content of an element
463 of this type is to be interpreted according to the definition of that format as provided outside of this
464 specification. If not otherwise indicated by the definition of the format, issues of anonymity,
465 pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties
466 are implementation-specific.

467 SPProvidedID [Optional]

468 A name identifier established by a service provider or affiliation of providers for the entity, if
469 different from the primary name identifier given in the content of the element. This attribute
470 provides a means of integrating the use of SAML with existing identifiers already in use by a
471 service provider. For example, an existing identifier can be "attached" to the entity using the Name
472 Identifier Management protocol defined in Section 3.6.

473 Additional rules for the content of (or the omission of) these attributes can be defined by elements that
474 make use of this type, and by specific `Format` definitions. The `NameQualifier` and `SPNameQualifier`
475 attributes SHOULD be omitted unless the element or format explicitly defines their use and semantics.

476 The following schema fragment defines the **NameIDType** complex type:

```
477 <complexType name="NameIDType">  
478   <simpleContent>
```

```

479     <extension base="string">
480         <attributeGroup ref="saml:IDNameQualifiers"/>
481         <attribute name="Format" type="anyURI" use="optional"/>
482         <attribute name="SPProvidedID" type="string" use="optional"/>
483     </extension>
484 </simpleContent>
485 </complexType>

```

486 2.2.3 Element <NameID>

487 The <NameID> element is of type **NameIDType** (see Section 2.2.2), and is used in various SAML
488 assertion constructs such as the <Subject> and <SubjectConfirmation> elements, and in various
489 protocol messages (see Section 3).

490 The following schema fragment defines the <NameID> element:

```

491 <element name="NameID" type="saml:NameIDType"/>

```

492 2.2.4 Element <EncryptedID>

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

496 <xenc:EncryptedData> [Required]

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

502 <xenc:EncryptedKey> [Zero or More]

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

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

510 Note that an entire assertion can be encrypted into this element and used as an identifier. In such a case,
511 the <Subject> element of the encrypted assertion supplies the "identifier" of the subject of the enclosing
512 assertion. Note also that if the identifying assertion is invalid, then so is the enclosing assertion.

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

```

515 <complexType name="EncryptedElementType">
516     <sequence>
517         <element ref="xenc:EncryptedData"/>
518         <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
519     </sequence>
520 </complexType>
521 <element name="EncryptedID" type="saml:EncryptedElementType"/>

```

522 2.2.5 Element <Issuer>

523 The <Issuer> element, with complex type **NameIDType**, provides information about the issuer of a
524 SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name,
525 but permits various pieces of descriptive data (see Section 2.2.2).

526 Overriding the usual rule for this element's type, if no `Format` value is provided with this element, then the
527 value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see Section 8.3.6).

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

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

530 2.3 Assertions

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

533 2.3.1 Element <AssertionIDRef>

534 The <AssertionIDRef> element makes a reference to a SAML assertion by its unique identifier. The
535 specific authority who issued the assertion or from whom the assertion can be obtained is not specified as
536 part of the reference. See Section 3.3.1 for a protocol element that uses such a reference to ask for the
537 corresponding assertion.

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

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

540 2.3.2 Element <AssertionURIRef>

541 The <AssertionURIRef> element makes a reference to a SAML assertion by URI reference. The URI
542 reference MAY be used to retrieve the corresponding assertion in a manner specific to the URI reference.
543 See Section 3.7 of the Bindings specification [SAMLBind] for information on how this element is used in a
544 protocol binding to accomplish this.

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

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

547 2.3.3 Element <Assertion>

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

550 `Version` [Required]

551 The version of this assertion. The identifier for the version of SAML defined in this specification is
552 "2.0". SAML versioning is discussed in Section 4.

553 `ID` [Required]

554 The identifier for this assertion. It is of type **xs:ID**, and MUST follow the requirements specified in
555 Section 1.3.4 for identifier uniqueness.

556 `IssueInstant` [Required]

557 The time instant of issue in UTC, as described in Section 1.3.3.

558 <Issuer> [Required]
559 The SAML authority that is making the claim(s) in the assertion. The issuer SHOULD be unambiguous
560 to the intended relying parties.

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

564 <ds:Signature> [Optional]
565 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as
566 described below and in Section 5.

567 <Subject> [Optional]
568 The subject of the statement(s) in the assertion.

569 <Conditions> [Optional]
570 Conditions that MUST be evaluated when assessing the validity of and/or when using the assertion.
571 See Section 2.5 for additional information on how to evaluate conditions.

572 <Advice> [Optional]
573 Additional information related to the assertion that assists processing in certain situations but which
574 MAY be ignored by applications that do not understand the advice or do not wish to make use of it.

575 Zero or more of the following statement elements:

576 <Statement>
577 A statement of a type defined in an extension schema. An `xsi:type` attribute MUST be used to
578 indicate the actual statement type.

579 <AuthnStatement>
580 An authentication statement.

581 <AuthzDecisionStatement>
582 An authorization decision statement.

583 <AttributeStatement>
584 An attribute statement.

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

588 Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If
589 <Subject> is omitted, then the statements in the assertion apply to a subject or subjects identified in an
590 application- or profile-specific manner. SAML itself defines no such statements, and an assertion without a
591 subject has no defined meaning in this specification.

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

597 If such a signature is used, then the <ds:Signature> element MUST be present, and a relying party
598 MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in
599 accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the
600 assertion. If it is valid, then the relying party SHOULD evaluate the signature to determine the identity and
601 appropriateness of the issuer and may continue to process the assertion in accordance with this

602 specification and as it deems appropriate (for example, evaluating conditions, advice, following profile-
603 specific rules, and so on).

604 Note that whether signed or unsigned, the inclusion of multiple statements within a single assertion is
605 semantically equivalent to a set of assertions containing those statements individually (provided the
606 subject, conditions, etc. are also the same).

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

```
608 <element name="Assertion" type="saml:AssertionType"/>
609 <complexType name="AssertionType">
610   <sequence>
611     <element ref="saml:Issuer"/>
612     <element ref="ds:Signature" minOccurs="0"/>
613     <element ref="saml:Subject" minOccurs="0"/>
614     <element ref="saml:Conditions" minOccurs="0"/>
615     <element ref="saml:Advice" minOccurs="0"/>
616     <choice minOccurs="0" maxOccurs="unbounded">
617       <element ref="saml:Statement"/>
618       <element ref="saml:AuthnStatement"/>
619       <element ref="saml:AuthzDecisionStatement"/>
620       <element ref="saml:AttributeStatement"/>
621     </choice>
622   </sequence>
623   <attribute name="Version" type="string" use="required"/>
624   <attribute name="ID" type="ID" use="required"/>
625   <attribute name="IssueInstant" type="dateTime" use="required"/>
626 </complexType>
```

627 2.3.4 Element <EncryptedAssertion>

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

631 <xenc:EncryptedData> [Required]

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

636 <xenc:EncryptedKey> [Zero or More]

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

641 Encrypted assertions are intended as a confidentiality protection mechanism when the plain-text value
642 passes through an intermediary.

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

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

645 2.4 Subjects

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

647 2.4.1 Element <Subject>

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

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

652 Identifies the subject.

653 <SubjectConfirmation> [Zero or More]

654 Information that allows the subject to be confirmed. If more than one subject confirmation is provided,
655 then satisfying any one of them is sufficient to confirm the subject for the purpose of applying the
656 assertion.

657 A <Subject> element can contain both an identifier and zero or more subject confirmations which a
658 relying party can verify when processing an assertion. If any one of the included subject confirmations are
659 verified, the relying party MAY treat the entity presenting the assertion as one that the asserting party has
660 associated with the principal identified in the name identifier and associated with the statements in the
661 assertion. This attesting entity and the actual subject may or may not be the same entity.

662 If there are no subject confirmations included, then any relationship between the presenter of the assertion
663 and the actual subject is unspecified.

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

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

```
666 <element name="Subject" type="saml:SubjectType"/>
667 <complexType name="SubjectType">
668   <choice>
669     <sequence>
670       <choice>
671         <element ref="saml:BaseID"/>
672         <element ref="saml:NameID"/>
673         <element ref="saml:EncryptedID"/>
674       </choice>
675       <element ref="saml:SubjectConfirmation" minOccurs="0"
676 maxOccurs="unbounded"/>
677     </sequence>
678     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
679   </choice>
680 </complexType>
```

681 2.4.1.1 Element <SubjectConfirmation>

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

685 Method [Required]

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

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

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

692 <SubjectConfirmationData> [Optional]

693 Additional confirmation information to be used by a specific confirmation method. For example, typical
694 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax
695 and Processing specification [XMLSig], which identifies a cryptographic key (See also Section
696 2.4.1.3). Particular confirmation methods MAY define a schema type to describe the elements,
697 attributes, or content that may appear in the <SubjectConfirmationData> element.

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

```
700 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
701 <complexType name="SubjectConfirmationType">
702   <sequence>
703     <choice minOccurs="0">
704       <element ref="saml:BaseID"/>
705       <element ref="saml:NameID"/>
706       <element ref="saml:EncryptedID"/>
707     </choice>
708     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
709   </sequence>
710   <attribute name="Method" type="anyURI" use="required"/>
711 </complexType>
```

712 2.4.1.2 Element <SubjectConfirmationData>

713 The <SubjectConfirmationData> element has the **SubjectConfirmationDataType** complex type. It
714 specifies additional data that allows the subject to be confirmed or constrains the circumstances under
715 which the act of subject confirmation can take place. Subject confirmation takes place when a relying
716 party seeks to verify the relationship between an entity presenting the assertion (that is, the attesting
717 entity) and the subject of the assertion's claims. It contains the following optional attributes that can apply
718 to any method:

719 **NotBefore** [Optional]

720 A time instant before which the subject cannot be confirmed. The time value is encoded in UTC, as
721 described in Section 1.3.3.

722 **NotOnOrAfter** [Optional]

723 A time instant at which the subject can no longer be confirmed. The time value is encoded in UTC, as
724 described in Section 1.3.3.

725 **Recipient** [Optional]

726 A URI specifying the entity or location to which an attesting entity can present the assertion. For
727 example, this attribute might indicate that the assertion must be delivered to a particular network
728 endpoint in order to prevent an intermediary from redirecting it someplace else.

729 **InResponseTo** [Optional]

730 The ID of a SAML protocol message in response to which an attesting entity can present the
731 assertion. For example, this attribute might be used to correlate the assertion to a SAML request that
732 resulted in its presentation.

733 **Address** [Optional]

734 The network address/location from which an attesting entity can present the assertion. For example,
735 this attribute might be used to bind the assertion to particular client addresses to prevent an attacker
736 from easily stealing and presenting the assertion from another location. IPv4 addresses SHOULD be
737 represented in the usual dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses SHOULD be
738 represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g.,
739 "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

740 Arbitrary attributes

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

748 Arbitrary elements

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

753 Particular confirmation methods and profiles that make use of those methods MAY require the use of one
754 or more of the attributes defined within this complex type. For examples of how these attributes (and
755 subject confirmation in general) can be used, see the Profiles specification [SAMLProf].

756 Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present,
757 SHOULD fall within the overall assertion validity period as specified by the `<Conditions>` element's
758 `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore`
759 MUST be less than (earlier than) the value for `NotOnOrAfter`.

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

```
762 <element name="SubjectConfirmationData"  
763 type="saml:SubjectConfirmationDataType"/>  
764 <complexType name="SubjectConfirmationDataType" mixed="true">  
765   <complexContent>  
766     <restriction base="anyType">  
767       <sequence>  
768         <any namespace="##any" processContents="lax" minOccurs="0"  
769 maxOccurs="unbounded"/>  
770       </sequence>  
771       <attribute name="NotBefore" type="dateTime" use="optional"/>  
772       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
773       <attribute name="Recipient" type="anyURI" use="optional"/>  
774       <attribute name="InResponseTo" type="NCName" use="optional"/>  
775       <attribute name="Address" type="string" use="optional"/>  
776       <anyAttribute namespace="##other" processContents="lax"/>  
777     </restriction>  
778   </complexContent>  
779 </complexType>
```

780 2.4.1.3 Complex Type KeyInfoConfirmationDataType

781 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`
782 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in
783 some way to authenticate an attesting entity. The particular confirmation method MUST define the exact
784 mechanism by which the confirmation data can be used. The optional attributes defined by the
785 **SubjectConfirmationDataType** complex type MAY also appear.

786 This complex type, or a type derived from it, SHOULD be used by any confirmation method that defines its
787 confirmation data in terms of the `<ds:KeyInfo>` element.

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

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

```
792 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
793   <complexContent>  
794     <restriction base="saml:SubjectConfirmationDataType">  
795       <sequence>  
796         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
797       </sequence>  
798     </restriction>  
799   </complexContent>  
800 </complexType>
```

801 2.4.1.4 Example of a Key-Confirmed `<Subject>`

802 To illustrate the way in which the various elements and types fit together, below is an example of a
803 `<Subject>` element containing a name identifier and a subject confirmation based on proof of
804 possession of a key. Note the use of the **KeyInfoConfirmationDataType** to identify the confirmation data
805 syntax as being a `<ds:KeyInfo>` element:

```
806 <Subject>  
807   <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">  
808     scott@example.org  
809   </NameID>  
810   <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">  
811     <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
812       <ds:KeyInfo>  
813         <ds:KeyName>Scott's Key</ds:KeyName>  
814       </ds:KeyInfo>  
815     </SubjectConfirmationData>  
816   </SubjectConfirmation>  
817 </Subject>
```

818 2.5 Conditions

819 This section defines the SAML constructs that place constraints on the acceptable use of SAML
820 assertions.

821 2.5.1 Element `<Conditions>`

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

823 `NotBefore` [Optional]

824 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as
825 described in Section 1.3.3.

826 `NotOnOrAfter` [Optional]

827 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as
828 described in Section 1.3.3.

829 `<Condition>` [Any Number]

830 A condition of a type defined in an extension schema. An `xsi:type` attribute MUST be used to
831 indicate the actual condition type.

832 `<AudienceRestriction>` [Any Number]

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

834 <OneTimeUse> [Optional]

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

838 <ProxyRestriction> [Optional]

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

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

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

```
849 <element name="Conditions" type="saml:ConditionsType"/>  
850 <complexType name="ConditionsType">  
851   <choice minOccurs="0" maxOccurs="unbounded">  
852     <element ref="saml:Condition"/>  
853     <element ref="saml:AudienceRestriction"/>  
854     <element ref="saml:OneTimeUse"/>  
855     <element ref="saml:ProxyRestriction"/>  
856   </choice>  
857   <attribute name="NotBefore" type="dateTime" use="optional"/>  
858   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
859 </complexType>
```

860 2.5.1.1 General Processing Rules

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

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

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

- 868 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is
869 considered to be **Valid** with respect to condition processing.
- 870 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the
871 assertion is considered to be **Invalid**.
- 872 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, or if an element is
873 encountered that is not understood, then the validity of the assertion cannot be determined and is
874 considered to be **Indeterminate**.
- 875 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the
876 assertion is considered to be **Valid** with respect to condition processing.

877 The first rule that applies terminates condition processing; thus a determination that an assertion is
878 **Invalid** takes precedence over that of **Indeterminate**.

879 An assertion that is determined to be **Invalid** or **Indeterminate** MUST be rejected by a relying party
880 (within whatever context or profile it was being processed), just as if the assertion were malformed or
881 otherwise unusable.

882 **2.5.1.2 Attributes NotBefore and NotOnOrAfter**

883 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion within
884 the context of its profile(s) of use. They do not guarantee that the statements in the assertion will be
885 correct or accurate throughout the validity period.

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

888 If the value for either `NotBefore` or `NotOnOrAfter` is omitted, then it is considered unspecified. If the
889 `NotBefore` attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), then
890 the assertion is **Valid** with respect to conditions at any time before the time instant specified by the
891 `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if all other conditions that are
892 supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions from the time instant specified
893 by the `NotBefore` attribute with no expiry. If neither attribute is specified (and if any other conditions that
894 are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions at any time.

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

897 **2.5.1.3 Element <Condition>**

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

900 The following schema fragment defines the `<Condition>` element and its **ConditionAbstractType**
901 complex type:

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

904 **2.5.1.4 Elements <AudienceRestriction> and <Audience>**

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

910 `<Audience>`

911 A URI reference that identifies an intended audience. The URI reference MAY identify a document
912 that describes the terms and conditions of audience membership. It MAY also contain the unique
913 identifier URI from a SAML name identifier that describes a system entity (see Section 8.3.6).

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

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

922 Note that multiple `<AudienceRestriction>` elements MAY be included in a single assertion, and each
923 MUST be evaluated independently. The effect of this requirement and the preceding definition is that
924 within a given condition, the audiences form a disjunction (an "OR") while multiple conditions form a
925 conjunction (an "AND").

926 The following schema fragment defines the `<AudienceRestriction>` element and its
927 **AudienceRestrictionType** complex type:

```
928 <element name="AudienceRestriction"  
929 type="saml:AudienceRestrictionType"/>  
930 <complexType name="AudienceRestrictionType">  
931   <complexContent>  
932     <extension base="saml:ConditionAbstractType">  
933       <sequence>  
934         <element ref="saml:Audience" maxOccurs="unbounded"/>  
935       </sequence>  
936     </extension>  
937   </complexContent>  
938 </complexType>  
939 <element name="Audience" type="anyURI"/>
```

940 2.5.1.5 Element `<OneTimeUse>`

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

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

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

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

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

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

964 The following schema fragment defines the `<OneTimeUse>` element and its **OneTimeUseType** complex
965 type:

```
966 <element name="OneTimeUse" type="saml:OneTimeUseType"/>  
967 <complexType name="OneTimeUseType">  
968   <complexContent>  
969     <extension base="saml:ConditionAbstractType"/>  
970   </complexContent>  
971 </complexType>
```


972 2.5.1.6 Element <ProxyRestriction>

973 Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as asserting
974 parties and issue subsequent assertions of their own on the basis of the information contained in the
975 original assertion. A relying party acting as an asserting party MUST NOT issue an assertion that itself
976 violates the restrictions specified in this condition on the basis of an assertion containing such a condition.

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

978 Count [Optional]

979 Specifies the maximum number of indirections that the asserting party permits to exist between this
980 assertion and an assertion which has ultimately been issued on the basis of it.

981 <Audience> [Zero or More]

982 Specifies the set of audiences to whom the asserting party permits new assertions to be issued on
983 the basis of this assertion.

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

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

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

994 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>
995 condition is considered to always be valid. That is, this condition does not affect validity but is a condition
996 on use.

997 The following schema fragment defines the <ProxyRestriction> element and its
998 **ProxyRestrictionType** complex type:

```
999 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
1000 <complexType name="ProxyRestrictionType">
1001   <complexContent>
1002     <extension base="saml:ConditionAbstractType">
1003       <sequence>
1004         <element ref="saml:Audience" minOccurs="0"
1005 maxOccurs="unbounded"/>
1006       </sequence>
1007       <attribute name="Count" type="nonNegativeInteger" use="optional"/>
1008     </extension>
1009   </complexContent>
1010 </complexType>
```

1011 2.6 Advice

1012 This section defines the SAML constructs that contain additional information about an assertion that an
1013 asserting party wishes to provide to a relying party.

1014 2.6.1 Element <Advice>

1015 The <Advice> element contains any additional information that the SAML authority wishes to provide.
1016 This information MAY be ignored by applications without affecting either the semantics or the validity of
1017 the assertion.

1018 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,
1019 <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in
1020 other non-SAML namespaces.

1021 Following are some potential uses of the <Advice> element:

- 1022 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating
1023 the claims) or indirectly (by reference to the supporting assertions).
- 1024 • State a proof of the assertion claims.
- 1025 • Specify the timing and distribution points for updates to the assertion.

1026 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
1027 <element name="Advice" type="saml:AdviceType"/>  
1028 <complexType name="AdviceType">  
1029   <choice minOccurs="0" maxOccurs="unbounded">  
1030     <element ref="saml:AssertionIDRef"/>  
1031     <element ref="saml:AssertionURIRef"/>  
1032     <element ref="saml:Assertion"/>  
1033     <element ref="saml:EncryptedAssertion"/>  
1034     <any namespace="##other" processContents="lax"/>  
1035   </choice>  
1036 </complexType>
```

1037 2.7 Statements

1038 The following sections define the SAML constructs that contain statement information.

1039 2.7.1 Element <Statement>

1040 The <Statement> element is an extension point that allows other assertion-based applications to reuse
1041 the SAML assertion framework. SAML itself derives its core statements from this extension point. Its
1042 **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

1043 The following schema fragment defines the <Statement> element and its **StatementAbstractType**
1044 complex type:

```
1045 <element name="Statement" type="saml:StatementAbstractType"/>  
1046 <complexType name="StatementAbstractType" abstract="true"/>
```

1047 2.7.2 Element <AuthnStatement>

1048 The <AuthnStatement> element describes a statement by the SAML authority asserting that the
1049 assertion subject was authenticated by a particular means at a particular time. Assertions containing
1050 <AuthnStatement> elements MUST contain a <Subject> element.

1051 It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the
1052 following elements and attributes:

1053 **Note:** The <AuthorityBinding> element and its corresponding type were removed
1054 from <AuthnStatement> for V2.0 of SAML.

1055 AuthnInstant [Required]

1056 Specifies the time at which the authentication took place. The time value is encoded in UTC, as
1057 described in Section 1.3.3.

1058 SessionIndex [Optional]

1059 Specifies the index of a particular session between the principal identified by the subject and the
1060 authenticating authority.

1061 SessionNotOnOrAfter [Optional]

1062 Specifies a time instant at which the session between the principal identified by the subject and the
1063 SAML authority issuing this statement MUST be considered ended. The time value is encoded in
1064 UTC, as described in Section 1.3.3. There is no required relationship between this attribute and a
1065 NotOnOrAfter condition attribute that may be present in the assertion.

1066 <SubjectLocality> [Optional]

1067 Specifies the DNS domain name and IP address for the system from which the assertion subject was
1068 apparently authenticated.

1069 <AuthnContext> [Required]

1070 The context used by the authenticating authority up to and including the authentication event that
1071 yielded this statement. Contains an authentication context class reference, an authentication context
1072 declaration or declaration reference, or both. See the Authentication Context specification
1073 [SAMLAuthnCxt] for a full description of authentication context information.

1074 In general, any string value MAY be used as a SessionIndex value. However, when privacy is a
1075 consideration, care must be taken to ensure that the SessionIndex value does not invalidate other
1076 privacy mechanisms. Accordingly, the value SHOULD NOT be usable to correlate activity by a principal
1077 across different session participants. Two solutions that achieve this goal are provided below and are
1078 RECOMMENDED:

- 1079 • Use small positive integers (or reoccurring constants in a list) for the SessionIndex. The SAML
1080 authority SHOULD choose the range of values such that the cardinality of any one integer will be
1081 sufficiently high to prevent a particular principal's actions from being correlated across multiple session
1082 participants. The SAML authority SHOULD choose values for SessionIndex randomly from within
1083 this range (except when required to ensure unique values for subsequent statements given to the
1084 same session participant but as part of a distinct session).
- 1085 • Use the enclosing assertion's ID value in the SessionIndex.

1086 The following schema fragment defines the <AuthnStatement> element and its AuthnStatementType
1087 complex type:

```
1088 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
1089 <complexType name="AuthnStatementType">
1090   <complexContent>
1091     <extension base="saml:StatementAbstractType">
1092       <sequence>
1093         <element ref="saml:SubjectLocality" minOccurs="0"/>
1094         <element ref="saml:AuthnContext"/>
1095       </sequence>
1096       <attribute name="AuthnInstant" type="dateTime" use="required"/>
1097       <attribute name="SessionIndex" type="string" use="optional"/>
1098       <attribute name="SessionNotOnOrAfter" type="dateTime"
1099 use="optional"/>
1100     </extension>
1101   </complexContent>
1102 </complexType>
```

1103 **2.7.2.1 Element <SubjectLocality>**

1104 The <SubjectLocality> element specifies the DNS domain name and IP address for the system from
1105 which the assertion subject was authenticated. It has the following attributes:

1106 Address [Optional]

1107 The network address of the system from which the principal identified by the subject was
1108 authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "1.2.3.4").
1109 IPv6 addresses SHOULD be represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513]
1110 (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

1111 DNSName [Optional]

1112 The DNS name of the system from which the principal identified by the subject was authenticated.

1113 This element is entirely advisory, since both of these fields are quite easily "spoofed," but may be useful
1114 information in some applications.

1115 The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**
1116 complex type:

```
1117 <element name="SubjectLocality" type="saml:SubjectLocalityType"/>  
1118 <complexType name="SubjectLocalityType">  
1119   <attribute name="Address" type="string" use="optional"/>  
1120   <attribute name="DNSName" type="string" use="optional"/>  
1121 </complexType>
```

1122 **2.7.2.2 Element <AuthnContext>**

1123 The <AuthnContext> element specifies the context of an authentication event. The element can contain
1124 an authentication context class reference, an authentication context declaration or declaration reference,
1125 or both. Its complex **AuthnContextType** has the following elements:

1126 <AuthnContextClassRef> [Optional]

1127 A URI reference identifying an authentication context class that describes the authentication context
1128 declaration that follows.

1129 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

1130 Either an authentication context declaration provided by value, or a URI reference that identifies such
1131 a declaration. The URI reference MAY directly resolve into an XML document containing the
1132 referenced declaration.

1133 <AuthenticatingAuthority> [Zero or More]

1134 Zero or more unique identifiers of authentication authorities that were involved in the authentication of
1135 the principal (not including the assertion issuer, who is presumed to have been involved without being
1136 explicitly named here).

1137 See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication
1138 context information.

1139 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**
1140 complex type:

```
1141 <element name="AuthnContext" type="saml:AuthnContextType"/>  
1142 <complexType name="AuthnContextType">  
1143   <sequence>  
1144     <choice>  
1145       <sequence>  
1146         <element ref="saml:AuthnContextClassRef"/>  
1147         <choice minOccurs="0">
```

```

1148         <element ref="saml:AuthnContextDecl"/>
1149         <element ref="saml:AuthnContextDeclRef"/>
1150     </choice>
1151 </sequence>
1152 <choice>
1153     <element ref="saml:AuthnContextDecl"/>
1154     <element ref="saml:AuthnContextDeclRef"/>
1155 </choice>
1156 </choice>
1157     <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1158 maxOccurs="unbounded"/>
1159 </sequence>
1160 </complexType>
1161 <element name="AuthnContextClassRef" type="anyURI"/>
1162 <element name="AuthnContextDeclRef" type="anyURI"/>
1163 <element name="AuthnContextDecl" type="anyType"/>
1164 <element name="AuthenticatingAuthority" type="anyURI"/>

```

1165 2.7.3 Element <AttributeStatement>

1166 The <AttributeStatement> element describes a statement by the SAML authority asserting that the
1167 assertion subject is associated with the specified attributes. Assertions containing
1168 <AttributeStatement> elements MUST contain a <Subject> element.

1169 It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the
1170 following elements:

1171 <Attribute> or <EncryptedAttribute> [One or More]

1172 The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML
1173 attribute may be included with the <EncryptedAttribute> element.

1174 The following schema fragment defines the <AttributeStatement> element and its
1175 **AttributeStatementType** complex type:

```

1176 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1177 <complexType name="AttributeStatementType">
1178     <complexContent>
1179         <extension base="saml:StatementAbstractType">
1180             <choice maxOccurs="unbounded">
1181                 <element ref="saml:Attribute"/>
1182                 <element ref="saml:EncryptedAttribute"/>
1183             </choice>
1184         </extension>
1185     </complexContent>
1186 </complexType>

```

1187 2.7.3.1 Element <Attribute>

1188 The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the
1189 **AttributeType** complex type. It is used within an attribute statement to express particular attributes and
1190 values associated with an assertion subject, as described in the previous section. It is also used in an
1191 attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.3 for
1192 more information). The <Attribute> element contains the following XML attributes:

1193 Name [Required]

1194 The name of the attribute.

1195 NameFormat [Optional]

1196 A URI reference representing the classification of the attribute name for purposes of interpreting the

1197 name. See Section 8.2 for some URI references that MAY be used as the value of the `NameFormat`
1198 attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided,
1199 the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section
1200 8.2.1) is in effect.

1201 `FriendlyName` [Optional]

1202 A string that provides a more human-readable form of the attribute's name, which may be useful in
1203 cases in which the actual `Name` is complex or opaque, such as an OID or a UUID. This attribute's
1204 value MUST NOT be used as a basis for formally identifying SAML attributes.

1205 Arbitrary attributes

1206 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to
1207 be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows
1208 additional fields to be added as needed to supply additional parameters to be used, for example, in an
1209 attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or
1210 XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a
1211 derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1212 `<AttributeValue>` [Any Number]

1213 Contains a value of the attribute. If an attribute contains more than one discrete value, it is
1214 RECOMMENDED that each value appear in its own `<AttributeValue>` element. If more than
1215 one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a
1216 datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have
1217 the identical datatype assigned.

1218 The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on
1219 its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the
1220 `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of
1221 values indicates that the requester is interested in any or all of the named attribute's values (see also
1222 Section 3.3.2.3).

1223 Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the
1224 semantics of specifying or omitting `<AttributeValue>` elements.

1225 The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
1226 <element name="Attribute" type="saml:AttributeType"/>
1227 <complexType name="AttributeType">
1228   <sequence>
1229     <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
1230   </sequence>
1231   <attribute name="Name" type="string" use="required"/>
1232   <attribute name="NameFormat" type="anyURI" use="optional"/>
1233   <attribute name="FriendlyName" type="string" use="optional"/>
1234   <anyAttribute namespace="##other" processContents="lax"/>
1235 </complexType>
```

1236 2.7.3.1.1 Element `<AttributeValue>`

1237 The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the
1238 **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

1239 If the data content of an `<AttributeValue>` element is of an XML Schema simple type (such as
1240 **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration
1241 in the `<AttributeValue>` element. If the attribute value contains structured data, the necessary data
1242 elements MAY be defined in an extension schema.

1243 **Note:** Specifying a datatype other than an XML Schema simple type on
1244 `<AttributeValue>` using `xsi:type` will require the presence of the extension schema
1245 that defines the datatype in order for schema processing to proceed.

1246 If a SAML attribute includes an empty value, such as the empty string, the corresponding
1247 `<AttributeValue>` element MUST be empty (generally this is serialized as `<AttributeValue/>`).
1248 This overrides the requirement in Section 1.3.1 that string values in SAML content contain at least one
1249 non-whitespace character.

1250 If a SAML attribute includes a "null" value, the corresponding `<AttributeValue>` element MUST be
1251 empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "true" or "1".

1252 The following schema fragment defines the `<AttributeValue>` element:

```
1253 <element name="AttributeValue" type="anyType" nillable="true"/>
```

1254 2.7.3.2 Element `<EncryptedAttribute>`

1255 The `<EncryptedAttribute>` element represents a SAML attribute in encrypted fashion, as defined by
1256 the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAttribute>`
1257 element contains the following elements:

1258 `<xenc:EncryptedData>` [Required]

1259 The encrypted content and associated encryption details, as defined by the XML Encryption
1260 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
1261 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
1262 encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

1263 `<xenc:EncryptedKey>` [Zero or More]

1264 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
1265 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
1266 the `Recipient` attribute SHOULD be the URI identifier of a system entity with a SAML name
1267 identifier, as defined by Section 8.3.6.

1268 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through
1269 an intermediary.

1270 The following schema fragment defines the `<EncryptedAttribute>` element:

```
1271 <element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

1272 2.7.4 Element `<AuthzDecisionStatement>`

1273 **Note:** The `<AuthzDecisionStatement>` feature has been frozen as of SAML V2.0,
1274 with no future enhancements planned. Users who require additional functionality may
1275 want to consider the eXtensible Access Control Markup Language [XACML], which offers
1276 enhanced authorization decision features.

1277 The `<AuthzDecisionStatement>` element describes a statement by the SAML authority asserting that
1278 a request for access by the assertion subject to the specified resource has resulted in the specified
1279 authorization decision on the basis of some optionally specified evidence. Assertions containing
1280 `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

1281 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1282 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
1283 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different

1284 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1285 URI references are to be found in IETF RFC 2396 [RFC 2396] Section 6:

1286 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1287 dependent. When a scheme uses elements of the common syntax, it will also use the common
1288 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1289 with an explicit ".port", where the port is the default for the scheme, is equivalent to one where
1290 the port is elided.

1291 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the
1292 URI normalized form wherever possible as follows:

- 1293 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1294 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1295 Inconsistent URI reference interpretation can also result from differences between the URI reference
1296 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1297 employed to specify an access control policy language. The following security conditions SHOULD be
1298 satisfied by the system which employs SAML assertions:

- 1299 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
1300 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
1301 part of the resource URI reference.
- 1302 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
1303 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
1304 gain access to a denied resource by creating such an equivalence.

1305 The `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends
1306 **StatementAbstractType** with the addition of the following elements and attributes:

1307 **Resource** [Required]

1308 A URI reference identifying the resource to which access authorization is sought. This attribute MAY
1309 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the
1310 current document", as specified by IETF RFC 2396 [RFC 2396] Section 4.2.

1311 **Decision** [Required]

1312 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1313 the **DecisionType** simple type.

1314 **<Action>** [One or more]

1315 The set of actions authorized to be performed on the specified resource.

1316 **<Evidence>** [Optional]

1317 A set of assertions that the SAML authority relied on in making the decision.

1318 The following schema fragment defines the `<AuthzDecisionStatement>` element and its
1319 **AuthzDecisionStatementType** complex type:

```
1320 <element name="AuthzDecisionStatement"  
1321 type="saml:AuthzDecisionStatementType"/>  
1322 <complexType name="AuthzDecisionStatementType">  
1323   <complexContent>  
1324     <extension base="saml:StatementAbstractType">  
1325       <sequence>  
1326         <element ref="saml:Action" maxOccurs="unbounded"/>  
1327         <element ref="saml:Evidence" minOccurs="0"/>  
1328       </sequence>  
1329       <attribute name="Resource" type="anyURI" use="required"/>  
1330       <attribute name="Decision" type="saml:DecisionType" use="required"/>  
1331     </extension>  
1332   </complexContent>  
</complexType>
```



```
1331     </extension>
1332   </complexContent>
1333 </complexType>
```

1334 2.7.4.1 Simple Type **DecisionType**

1335 The **DecisionType** simple type defines the possible values to be reported as the status of an
1336 authorization decision statement.

1337 Permit

1338 The specified action is permitted.

1339 Deny

1340 The specified action is denied.

1341 Indeterminate

1342 The SAML authority cannot determine whether the specified action is permitted or denied.

1343 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to
1344 provide an affirmative statement but where it is not able to issue a decision. Additional information as to
1345 the reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>`
1346 elements in the enclosing `<Response>`.

1347 The following schema fragment defines the **DecisionType** simple type:

```
1348 <simpleType name="DecisionType">
1349   <restriction base="string">
1350     <enumeration value="Permit"/>
1351     <enumeration value="Deny"/>
1352     <enumeration value="Indeterminate"/>
1353   </restriction>
1354 </simpleType>
```

1355 2.7.4.2 Element **<Action>**

1356 The `<Action>` element specifies an action on the specified resource for which permission is sought. Its
1357 string-data content provides the label for an action sought to be performed on the specified resource, and
1358 it has the following attribute:

1359 Namespace [Optional]

1360 A URI reference representing the namespace in which the name of the specified action is to be
1361 interpreted. If this element is absent, the namespace
1362 `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation` specified in Section 8.1.2 is in
1363 effect.

1364 The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```
1365 <element name="Action" type="saml:ActionType"/>
1366 <complexType name="ActionType">
1367   <simpleContent>
1368     <extension base="string">
1369       <attribute name="Namespace" type="anyURI" use="required"/>
1370     </extension>
1371   </simpleContent>
1372 </complexType>
```

1373 2.7.4.3 Element <Evidence>

1374 The <Evidence> element contains one or more assertions or assertion references that the SAML
1375 authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains
1376 a mixture of one or more of the following elements:

1377 <AssertionIDRef> [Any number]

1378 Specifies an assertion by reference to the value of the assertion's ID attribute.

1379 <AssertionURIRef> [Any number]

1380 Specifies an assertion by means of a URI reference.

1381 <Assertion> [Any number]

1382 Specifies an assertion by value.

1383 <EncryptedAssertion> [Any number]

1384 Specifies an encrypted assertion by value.

1385 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1386 and the SAML authority making the authorization decision. For example, in the case that the SAML relying
1387 party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1388 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1389 assertion as valid either to the relying party or any other third party.

1390 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
1391 <element name="Evidence" type="saml:EvidenceType"/>  
1392 <complexType name="EvidenceType">  
1393   <choice maxOccurs="unbounded">  
1394     <element ref="saml:AssertionIDRef"/>  
1395     <element ref="saml:AssertionURIRef"/>  
1396     <element ref="saml:Assertion"/>  
1397     <element ref="saml:EncryptedAssertion"/>  
1398   </choice>  
1399 </complexType>
```

3 SAML Protocols

1400

1401 SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML
1402 bindings specification [SAMLBind] describes specific means of transporting protocol messages using
1403 existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a
1404 number of applications of the protocols defined in this section together with additional processing rules,
1405 restrictions, and requirements that facilitate interoperability.

1406 Specific SAML request and response messages derive from common types. The requester sends an
1407 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1408 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1409



1411 Figure 1: SAML Request-Response Protocol

1412 In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the
1413 responder having received a corresponding request.

1414 The protocols defined by SAML achieve the following actions:

- 1415 • Returning one or more requested assertions. This can occur in response to either a direct request
1416 for specific assertions or a query for assertions that meet particular criteria.
- 1417 • Performing authentication on request and returning the corresponding assertion
- 1418 • Registering a name identifier or terminating a name registration on request
- 1419 • Retrieving a protocol message that has been requested by means of an artifact
- 1420 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
1421 request
- 1422 • Providing a name identifier mapping on request

1423 Throughout this section, text descriptions of elements and types in the SAML protocol namespace are not
1424 shown with the conventional namespace prefix `samlp:`. For clarity, text descriptions of elements and
1425 types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:`.

3.1 Schema Header and Namespace Declarations

1426

1427 The following schema fragment defines the XML namespaces and other header information for the
1428 protocol schema:

1429

```
1430 <schema  
1431   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1432   xmlns="http://www.w3.org/2001/XMLSchema"  
1433   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1434   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1435   xmlns:ds="http://www.w3.org/2000/09/xmldsig#" elementFormDefault="unqualified"  
1436   attributeFormDefault="unqualified"  
1437   blockDefault="substitution"  
1438   version="2.0">
```

```

1439 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1440       schemaLocation="saml-schema-assertion-2.0.xsd"/>
1441 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1442       schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
1443 20020212/xmldsig-core-schema.xsd"/>
1444 <annotation>
1445   <documentation>
1446     Document identifier: saml-schema-protocol-2.0
1447     Location: http://docs.oasis-open.org/security/saml/v2.0/
1448     Revision history:
1449     V1.0 (November, 2002):
1450       Initial Standard Schema.
1451     V1.1 (September, 2003):
1452       Updates within the same V1.0 namespace.
1453     V2.0 (March, 2005):
1454       New protocol schema based in a SAML V2.0 namespace.
1455   </documentation>
1456 </annotation>
1457 ...
1458 </schema>

```

1459 3.2 Requests and Responses

1460 The following sections define the SAML constructs and basic requirements that underlie all of the request
1461 and response messages used in SAML protocols.

1462 3.2.1 Complex Type RequestAbstractType

1463 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1464 This type defines common attributes and elements that are associated with all SAML requests:

1465 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**
1466 for V2.0 of SAML.

1467 ID [Required]

1468 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section
1469 1.3.4 for identifier uniqueness. The values of the ID attribute in a request and the InResponseTo
1470 attribute in the corresponding response MUST match.

1471 Version [Required]

1472 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".
1473 SAML versioning is discussed in Section 4.

1474 IssueInstant [Required]

1475 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
1476 1.3.3.

1477 Destination [Optional]

1478 A URI reference indicating the address to which this request has been sent. This is useful to prevent
1479 malicious forwarding of requests to unintended recipients, a protection that is required by some
1480 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1481 location at which the message was received. If it does not, the request MUST be discarded. Some
1482 protocol bindings may require the use of this attribute (see [SAMLBind]).

1483 Consent [Optional]

1484 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1485 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value

1486 of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the
1487 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in
1488 effect.

1489 `<saml:Issuer>` [Optional]

1490 Identifies the entity that generated the request message. (For more information on this element, see
1491 Section 2.2.5.)

1492 `<ds:Signature>` [Optional]

1493 An XML Signature that authenticates the requester and provides message integrity, as described
1494 below and in Section 5.

1495 `<Extensions>` [Optional]

1496 This extension point contains optional protocol message extension elements that are agreed on
1497 between the communicating parties. No extension schema is required in order to make use of this
1498 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1499 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1500 SAML-defined namespace.

1501 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to
1502 authenticate itself, and message integrity may often be required. Authentication and message integrity
1503 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request
1504 MAY be signed, which provides both authentication of the requester and message integrity.

1505 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML
1506 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)
1507 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the
1508 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the
1509 signature to determine the identity and appropriateness of the signer and may continue to process the
1510 request or respond with an error (if the request is invalid for some other reason).

1511 If a `Consent` attribute is included and the value indicates that some form of principal consent has been
1512 obtained, then the request SHOULD be signed.

1513 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if
1514 it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value
1515 `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a
1516 suspected denial-of-service attack, not responding at all may be warranted.

1517 The following schema fragment defines the **RequestAbstractType** complex type:

```
1518 <complexType name="RequestAbstractType" abstract="true">
1519   <sequence>
1520     <element ref="saml:Issuer" minOccurs="0"/>
1521     <element ref="ds:Signature" minOccurs="0"/>
1522     <element ref="samlp:Extensions" minOccurs="0"/>
1523   </sequence>
1524   <attribute name="ID" type="ID" use="required"/>
1525   <attribute name="Version" type="string" use="required"/>
1526   <attribute name="IssueInstant" type="dateTime" use="required"/>
1527   <attribute name="Destination" type="anyURI" use="optional"/>
1528   <attribute name="Consent" type="anyURI" use="optional"/>
1529 </complexType>
1530 <element name="Extensions" type="samlp:ExtensionsType"/>
1531 <complexType name="ExtensionsType">
1532   <sequence>
1533     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1534   </sequence>
1535 </complexType>
```

1536 3.2.2 Complex Type StatusResponseType

1537 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
1538 defines common attributes and elements that are associated with all SAML responses:

1539 ID [Required]

1540 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in
1541 Section 1.3.4 for identifier uniqueness.

1542 InResponseTo [Optional]

1543 A reference to the identifier of the request to which the response corresponds, if any. If the response
1544 is not generated in response to a request, or if the ID attribute value of a request cannot be
1545 determined (for example, the request is malformed), then this attribute MUST NOT be present.
1546 Otherwise, it MUST be present and its value MUST match the value of the corresponding request's
1547 ID attribute.

1548 Version [Required]

1549 The version of this response. The identifier for the version of SAML defined in this specification is
1550 "2.0". SAML versioning is discussed in Section 4.

1551 IssueInstant [Required]

1552 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
1553 1.3.3.

1554 Destination [Optional]

1555 A URI reference indicating the address to which this response has been sent. This is useful to prevent
1556 malicious forwarding of responses to unintended recipients, a protection that is required by some
1557 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1558 location at which the message was received. If it does not, the response MUST be discarded. Some
1559 protocol bindings may require the use of this attribute (see [SAMLBind]).

1560 Consent [Optional]

1561 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1562 the sending of this response. See Section 8.4 for some URI references that MAY be used as the value
1563 of the Consent attribute and their associated descriptions. If no Consent value is provided, the
1564 identifier urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in
1565 effect.

1566 <saml:Issuer> [Optional]

1567 Identifies the entity that generated the response message. (For more information on this element, see
1568 Section 2.2.5.)

1569 <ds:Signature> [Optional]

1570 An XML Signature that authenticates the responder and provides message integrity, as described
1571 below and in Section 5.

1572 <Extensions> [Optional]

1573 This extension point contains optional protocol message extension elements that are agreed on
1574 between the communicating parties. . No extension schema is required in order to make use of this
1575 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1576 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1577 SAML-defined namespace.

1578 <Status> [Required]

1579 A code representing the status of the corresponding request.

1580 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to
1581 authenticate itself, and message integrity may often be required. Authentication and message integrity
1582 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML
1583 response MAY be signed, which provides both authentication of the responder and message integrity.

1584 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML
1585 requester receiving the response MUST verify that the signature is valid (that is, that the message has not
1586 been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on
1587 the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD
1588 evaluate the signature to determine the identity and appropriateness of the signer and may continue to
1589 process the response as it deems appropriate.

1590 If a `Consent` attribute is included and the value indicates that some form of principal consent has been
1591 obtained, then the response SHOULD be signed.

1592 The following schema fragment defines the **StatusResponseType** complex type:

```
1593 <complexType name="StatusResponseType">  
1594   <sequence>  
1595     <element ref="saml:Issuer" minOccurs="0"/>  
1596     <element ref="ds:Signature" minOccurs="0"/>  
1597     <element ref="samlp:Extensions" minOccurs="0"/>  
1598     <element ref="samlp:Status"/>  
1599   </sequence>  
1600   <attribute name="ID" type="ID" use="required"/>  
1601   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1602   <attribute name="Version" type="string" use="required"/>  
1603   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1604   <attribute name="Destination" type="anyURI" use="optional"/>  
1605   <attribute name="Consent" type="anyURI" use="optional"/>  
1606 </complexType>
```

1607 3.2.2.1 Element `<Status>`

1608 The `<Status>` element contains the following elements:

1609 `<StatusCode>` [Required]

1610 A code representing the status of the activity carried out in response to the corresponding request.

1611 `<StatusMessage>` [Optional]

1612 A message which MAY be returned to an operator.

1613 `<StatusDetail>` [Optional]

1614 Additional information concerning the status of the request.

1615 The following schema fragment defines the `<Status>` element and its **StatusType** complex type:

```
1616 <element name="Status" type="samlp:StatusType"/>  
1617 <complexType name="StatusType">  
1618   <sequence>  
1619     <element ref="samlp:StatusCode"/>  
1620     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1621     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1622   </sequence>  
1623 </complexType>
```

1624 3.2.2.2 Element `<StatusCode>`

1625 The `<StatusCode>` element specifies a code or a set of nested codes representing the status of the
1626 corresponding request. The `<StatusCode>` element has the following element and attribute:

1627 Value [Required]

1628 The status code value. This attribute contains a URI reference. The value of the topmost
1629 <StatusCode> element MUST be from the top-level list provided in this section.

1630 <StatusCode> [Optional]

1631 A subordinate status code that provides more specific information on an error condition. Note that
1632 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for
1633 additional information by intentionally presenting erroneous requests.

1634 The permissible top-level <StatusCode> values are as follows:

1635 urn:oasis:names:tc:SAML:2.0:status:Success

1636 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1637 <StatusDetail> elements.

1638 urn:oasis:names:tc:SAML:2.0:status:Requester

1639 The request could not be performed due to an error on the part of the requester.

1640 urn:oasis:names:tc:SAML:2.0:status:Responder

1641 The request could not be performed due to an error on the part of the SAML responder or SAML
1642 authority.

1643 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1644 The SAML responder could not process the request because the version of the request message was
1645 incorrect.

1646 The following second-level status codes are referenced at various places in this specification. Additional
1647 second-level status codes MAY be defined in future versions of the SAML specification. System entities
1648 are free to define more specific status codes by defining appropriate URI references.

1649 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1650 The responding provider was unable to successfully authenticate the principal.

1651 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1652 Unexpected or invalid content was encountered within a <saml:Attribute> or
1653 <saml:AttributeValue> element.

1654 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1655 The responding provider cannot or will not support the requested name identifier policy.

1656 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1657 The specified authentication context requirements cannot be met by the responder.

1658 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1659 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an
1660 <IDPList> can be resolved or that none of the supported identity providers are available.

1661 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1662 Indicates the responding provider cannot authenticate the principal passively, as has been requested.

1663 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP

1664 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1665 supported by the intermediary.

1666 urn:oasis:names:tc:SAML:2.0:status:PartialLogout
 1667 Used by a session authority to indicate to a session participant that it was not able to propagate logout
 1668 to all other session participants.

1669 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
 1670 Indicates that a responding provider cannot authenticate the principal directly and is not permitted to
 1671 proxy the request further.

1672 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
 1673 The SAML responder or SAML authority is able to process the request but has chosen not to respond.
 1674 This status code MAY be used when there is concern about the security context of the request
 1675 message or the sequence of request messages received from a particular requester.

1676 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
 1677 The SAML responder or SAML authority does not support the request.

1678 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
 1679 The SAML responder cannot process any requests with the protocol version specified in the request.

1680 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
 1681 The SAML responder cannot process the request because the protocol version specified in the
 1682 request message is a major upgrade from the highest protocol version supported by the responder.

1683 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
 1684 The SAML responder cannot process the request because the protocol version specified in the
 1685 request message is too low.

1686 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
 1687 The resource value provided in the request message is invalid or unrecognized.

1688 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
 1689 The response message would contain more elements than the SAML responder is able to return.

1690 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
 1691 An entity that has no knowledge of a particular attribute profile has been presented with an attribute
 1692 drawn from that profile.

1693 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
 1694 The responding provider does not recognize the principal specified or implied by the request.

1695 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
 1696 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
 1697 request.

1698 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
 1699 type:

```

1700 <element name="StatusCode" type="samlp:StatusCodeType"/>
1701 <complexType name="StatusCodeType">
1702   <sequence>
1703     <element ref="samlp:StatusCode" minOccurs="0"/>
1704   </sequence>
1705   <attribute name="Value" type="anyURI" use="required"/>
1706 </complexType>

```

1707 3.2.2.3 Element <StatusMessage>

1708 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1709 The following schema fragment defines the <StatusMessage> element:

```
1710 <element name="StatusMessage" type="string"/>
```

1711 3.2.2.4 Element <StatusDetail>

1712 The <StatusDetail> element MAY be used to specify additional information concerning the status of
1713 the request. The additional information consists of zero or more elements from any namespace, with no
1714 requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

1715 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1716 complex type:

```
1717 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1718 <complexType name="StatusDetailType">  
1719   <sequence>  
1720     <any namespace="##any" processContents="lax" minOccurs="0"  
1721     maxOccurs="unbounded"/>  
1722   </sequence>  
1723 </complexType>
```

1724 3.3 Assertion Query and Request Protocol

1725 This section defines messages and processing rules for requesting existing assertions by reference or
1726 querying for assertions by subject and statement type.

1727 3.3.1 Element <AssertionIDRequest>

1728 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>
1729 message element can be used to request that they be returned in a <Response> message. The
1730 <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for
1731 more information on this element.

1732 The following schema fragment defines the <AssertionIDRequest> element:

```
1733 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>  
1734 <complexType name="AssertionIDRequestType">  
1735   <complexContent>  
1736     <extension base="samlp:RequestAbstractType">  
1737       <sequence>  
1738         <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>  
1739       </sequence>  
1740     </extension>  
1741   </complexContent>  
1742 </complexType>
```

1743 3.3.2 Queries

1744 The following sections define the SAML query request messages.

1745 3.3.2.1 Element <SubjectQuery>

1746 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1747 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and

1748 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the
1749 <saml:Subject> element (defined in Section 2.4) to **RequestAbstractType**.

1750 The following schema fragment defines the <SubjectQuery> element and its
1751 **SubjectQueryAbstractType** complex type:

```
1752 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1753 <complexType name="SubjectQueryAbstractType" abstract="true">  
1754   <complexContent>  
1755     <extension base="samlp:RequestAbstractType">  
1756       <sequence>  
1757         <element ref="saml:Subject"/>  
1758       </sequence>  
1759     </extension>  
1760   </complexContent>  
1761 </complexType>
```

1762 3.3.2.2 Element <AuthnQuery>

1763 The <AuthnQuery> message element is used to make the query “What assertions containing
1764 authentication statements are available for this subject?” A successful <Response> will contain one or
1765 more assertions containing authentication statements.

1766 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1767 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1768 that have occurred in a previous interaction between the indicated subject and the authentication authority.

1769 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1770 the following element and attribute:

1771 SessionIndex [Optional]

1772 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1773 containing authentication statements do you have for this subject within the context of the supplied
1774 session information?”

1775 <RequestedAuthnContext> [Optional]

1776 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1777 containing authentication statements do you have for this subject that satisfy the authentication
1778 context requirements in this element?”

1779 In response to an authentication query, a SAML authority returns assertions with authentication
1780 statements as follows:

- 1781 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1782 assertions that may be returned.
- 1783 • If the SessionIndex attribute is present in the query, at least one <AuthnStatement> element in
1784 the set of returned assertions MUST contain a SessionIndex attribute that matches the
1785 SessionIndex attribute in the query. It is OPTIONAL for the complete set of all such matching
1786 assertions to be returned in the response.
- 1787 • If the <RequestedAuthnContext> element is present in the query, at least one
1788 <AuthnStatement> element in the set of returned assertions MUST contain an
1789 <AuthnContext> element that satisfies the element in the query (see Section 3.3.2.2.1). It is
1790 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1791 The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex
1792 type:

```
1793 <element name="AuthnQuery" type="samlp:AuthnQueryType"/>
```

```

1794 <complexType name="AuthnQueryType">
1795   <complexContent>
1796     <extension base="samlp:SubjectQueryAbstractType">
1797       <sequence>
1798         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
1799       </sequence>
1800       <attribute name="SessionIndex" type="string" use="optional"/>
1801     </extension>
1802   </complexContent>
1803 </complexType>

```

1804 3.3.2.2.1 Element <RequestedAuthnContext>

1805 The <RequestedAuthnContext> element specifies the authentication context requirements of
 1806 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
 1807 complex type defines the following elements and attributes:

1808 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1809 Specifies one or more URI references identifying authentication context classes or declarations.
 1810 These elements are defined in Section 2.7.2.2. For more information about authentication context
 1811 classes, see [SAMLAuthnCxt].

1812 Comparison [Optional]

1813 Specifies the comparison method used to evaluate the requested context classes or statements, one
 1814 of "exact", "minimum", "maximum", or "better". The default is "exact".

1815 Either a set of class references or a set of declaration references can be used. The set of supplied
 1816 references **MUST** be evaluated as an ordered set, where the first element is the most preferred
 1817 authentication context class or declaration. If none of the specified classes or declarations can be satisfied
 1818 in accordance with the rules below, then the responder **MUST** return a <Response> message with a
 1819 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1820 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
 1821 statement **MUST** be the exact match of at least one of the authentication contexts specified.

1822 If Comparison is set to "minimum", then the resulting authentication context in the authentication
 1823 statement **MUST** be at least as strong (as deemed by the responder) as one of the authentication
 1824 contexts specified.

1825 If Comparison is set to "better", then the resulting authentication context in the authentication
 1826 statement **MUST** be stronger (as deemed by the responder) than any one of the authentication contexts
 1827 specified.

1828 If Comparison is set to "maximum", then the resulting authentication context in the authentication
 1829 statement **MUST** be as strong as possible (as deemed by the responder) without exceeding the strength
 1830 of at least one of the authentication contexts specified.

1831 The following schema fragment defines the <RequestedAuthnContext> element and its
 1832 **RequestedAuthnContextType** complex type:

```

1833 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1834 <complexType name="RequestedAuthnContextType">
1835   <choice>
1836     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1837     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1838   </choice>
1839   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1840 use="optional"/>
1841 </complexType>
1842 <simpleType name="AuthnContextComparisonType">

```

```

1843     <restriction base="string">
1844         <enumeration value="exact"/>
1845         <enumeration value="minimum"/>
1846         <enumeration value="maximum"/>
1847         <enumeration value="better"/>
1848     </restriction>
1849 </simpleType>

```

1850 3.3.2.3 Element <AttributeQuery>

1851 The <AttributeQuery> element is used to make the query “Return the requested attributes for this
1852 subject.” A successful response will be in the form of assertions containing attribute statements, to the
1853 extent allowed by policy. This element is of type **AttributeQueryType**, which extends
1854 **SubjectQueryAbstractType** with the addition of the following element:

1855 <saml:Attribute> [Any Number]

1856 Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no
1857 attributes are specified, it indicates that all attributes allowed by policy are requested. If a given
1858 <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if
1859 that attribute is returned in the response, it MUST NOT contain any values that are not equal to the
1860 values specified in the query. In the absence of equality rules specified by particular profiles or
1861 attributes, equality is defined as an identical XML representation of the value. For more information on
1862 <saml:Attribute>, see Section 2.7.3.1.

1863 A single query MUST NOT contain two <saml:Attribute> elements with the same Name and
1864 NameFormat values (that is, a given attribute MUST be named only once in a query).

1865 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1866 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1867 assertions that may be returned.
- 1868 • If any <Attribute> elements are present in the query, they constrain/filter the attributes and
1869 optionally the values returned, as noted above.
- 1870 • The attributes and values returned MAY also be constrained by application-specific policy
1871 considerations.

1872 The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1873 and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to
1874 indicate problems with the interpretation of attribute or value information in a query.

1875 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1876 complex type:

```

1877 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1878 <complexType name="AttributeQueryType">
1879     <complexContent>
1880         <extension base="samlp:SubjectQueryAbstractType">
1881             <sequence>
1882                 <element ref="saml:Attribute" minOccurs="0"
1883 maxOccurs="unbounded"/>
1884             </sequence>
1885         </extension>
1886     </complexContent>
1887 </complexType>

```

1888 3.3.2.4 Element <AuthzDecisionQuery>

1889 The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource
1890 be allowed for this subject, given this evidence?” A successful response will be in the form of assertions
1891 containing authorization decision statements.

1892 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no
1893 future enhancements planned. Users who require additional functionality may want to
1894 consider the eXtensible Access Control Markup Language [XACML], which offers
1895 enhanced authorization decision features.

1896 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1897 addition of the following elements and attribute:

1898 Resource [Required]

1899 A URI reference indicating the resource for which authorization is requested.

1900 <saml:Action> [One or More]

1901 The actions for which authorization is requested. For more information on this element, see Section
1902 2.7.4.2.

1903 <saml:Evidence> [Optional]

1904 A set of assertions that the SAML authority MAY rely on in making its authorization decision. For more
1905 information on this element, see Section 2.7.4.3.

1906 In response to an authorization decision query, a SAML authority returns assertions with authorization
1907 decision statements as follows:

- 1908 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1909 assertions that may be returned.

1910 The following schema fragment defines the <AuthzDecisionQuery> element and its
1911 **AuthzDecisionQueryType** complex type:

```
1912 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>  
1913 <complexType name="AuthzDecisionQueryType">  
1914   <complexContent>  
1915     <extension base="samlp:SubjectQueryAbstractType">  
1916       <sequence>  
1917         <element ref="saml:Action" maxOccurs="unbounded"/>  
1918         <element ref="saml:Evidence" minOccurs="0"/>  
1919       </sequence>  
1920       <attribute name="Resource" type="anyURI" use="required"/>  
1921     </extension>  
1922   </complexContent>  
1923 </complexType>
```

1924 3.3.3 Element <Response>

1925 The <Response> message element is used when a response consists of a list of zero or more assertions
1926 that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType**
1927 and adds the following elements:

1928 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1929 Specifies an assertion by value, or optionally an encrypted assertion by value. See Section 2.3.3 for
1930 more information on these elements.

1931 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1932 <element name="Response" type="samlp:ResponseType"/>
1933 <complexType name="ResponseType">
1934   <complexContent>
1935     <extension base="samlp:StatusResponseType">
1936       <choice minOccurs="0" maxOccurs="unbounded">
1937         <element ref="saml:Assertion"/>
1938         <element ref="saml:EncryptedAssertion"/>
1939       </choice>
1940     </extension>
1941   </complexContent>
1942 </complexType>

```

1943 3.3.4 Processing Rules

1944 In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST**
 1945 contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the
 1946 query.

1947 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both
 1948 apply:

- 1949 • If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST**
 1950 include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2.
 1951 In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical"
 1952 means that the identifier element's content and attribute values **MUST** be the same. An encrypted
 1953 identifier will be identical to the original according to this definition, once decrypted.
- 1954 • If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at
 1955 least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner
 1956 described by at least one `<saml:SubjectConfirmation>` element in S2.

1957 As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular
 1958 Format value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting
 1959 S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with
 1960 different Format values and element content, even if the two identifiers are considered to refer to the
 1961 same principal.

1962 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
 1963 expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an
 1964 `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value
 1965 `urn:oasis:names:tc:SAML:2.0:status:Success`.

1966 All other processing rules associated with the underlying request and response messages **MUST** be
 1967 observed.

1968 3.4 Authentication Request Protocol

1969 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
 1970 authentication statements to establish a security context at one or more relying parties, it can use the
 1971 authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and
 1972 request that it return a `<Response>` message containing one or more such assertions. Such assertions
 1973 **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one
 1974 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

1975 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
 1976 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
 1977 is not specified, though the means of authentication might impact the content of the response. Other
 1978 issues related to the validation of authentication credentials by the identity provider or any communication

1979 between the identity provider and any other entities involved in the authentication process are also out of
1980 scope of this protocol.

1981 The descriptions and processing rules in the following sections reference the following actors, many of
1982 whom might be the same entity in a particular profile of use:

1983 Requester

1984 The entity who creates the authentication request and to whom the response is to be returned.

1985 Presenter

1986 The entity who presents the request to the identity provider and either authenticates itself during
1987 the transmission of the message, or relies on an existing security context to establish its identity. If
1988 not the requester, the presenter acts as an intermediary between the requester and the
1989 responding identity provider.

1990 Requested Subject

1991 The entity about whom one or more assertions are being requested.

1992 Attesting Entity

1993 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`
1994 elements of the resulting assertion(s).

1995 Relying Party

1996 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
1997 the profile or context of use, generally to establish a security context.

1998 Identity Provider

1999 The entity to whom the presenter gives the request and from whom the presenter receives the
2000 response.

2001 **3.4.1 Element `<AuthnRequest>`**

2002 To request that an identity provider issue an assertion with an authentication statement, a presenter
2003 authenticates to that identity provider (or relies on an existing security context) and sends it an
2004 `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to
2005 satisfy its purpose. Among these properties may be information that relates to the content of the assertion
2006 and/or information that relates to how the resulting `<Response>` message should be delivered to the
2007 requester. The process of authentication of the presenter may take place before, during, or after the initial
2008 delivery of the `<AuthnRequest>` message.

2009 The requester might not be the same as the presenter of the request if, for example, the requester is a
2010 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject
2011 so that the relying party can decide whether to provide a service.

2012 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2013 by the protocol binding used to deliver the message.

2014 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
2015 adds the following elements and attributes, all of which are optional in general, but may be required by
2016 specific profiles:

2017 `<saml:Subject>` [Optional]

2018 Specifies the requested subject of the resulting assertion(s). This may include one or more
2019 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions
2020 can be confirmed. For more information on this element, see Section 2.4.

2021 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
2022 requested subject. If no <saml:SubjectConfirmation> elements are included, then the presenter
2023 is presumed to be the only attesting entity required and the method is implied by the profile of use
2024 and/or the policies of the identity provider.

2025 <NameIDPolicy> [Optional]

2026 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
2027 then any type of identifier supported by the identity provider for the requested subject can be used,
2028 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2029 <saml:Conditions> [Optional]

2030 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting
2031 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
2032 information in this element is used as input to the process of constructing the assertion, rather than as
2033 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2034 <RequestedAuthnContext> [Optional]

2035 Specifies the requirements, if any, that the requester places on the authentication context that applies
2036 to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for processing rules
2037 regarding this element.

2038 <Scoping> [Optional]

2039 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as
2040 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
2041 providers by the responder.

2042 ForceAuthn [Optional]

2043 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
2044 rely on a previous security context. If a value is not provided, the default is "false". However, if both
2045 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
2046 presenter unless the constraints of IsPassive can be met.

2047 IsPassive [Optional]

2048 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take control
2049 of the user interface from the requester and interact with the presenter in a noticeable fashion. If a
2050 value is not provided, the default is "false".

2051 AssertionConsumerServiceIndex [Optional]

2052 Indirectly identifies the location to which the <Response> message should be returned to the
2053 requester. It applies only to profiles in which the requester is different from the presenter, such as the
2054 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map
2055 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one
2056 possible mechanism. If omitted, then the identity provider MUST return the <Response> message to
2057 the default location associated with the requester for the profile of use. If the index specified is invalid,
2058 then the identity provider MAY return an error <Response> or it MAY use the default location. This
2059 attribute is mutually exclusive with the AssertionConsumerServiceURL and ProtocolBinding
2060 attributes.

2061 AssertionConsumerServiceURL [Optional]

2062 Specifies by value the location to which the <Response> message MUST be returned to the
2063 requester. The responder MUST ensure by some means that the value specified is in fact associated
2064 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing
2065 <AuthnRequest> message is another. This attribute is mutually exclusive with the
2066 AssertionConsumerServiceIndex attribute and is typically accompanied by the
2067 ProtocolBinding attribute.

2068 ProtocolBinding [Optional]

2069 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>
2070 message. See [SAMLBind] for more information about protocol bindings and URI references defined
2071 for them. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute
2072 and is typically accompanied by the AssertionConsumerServiceURL attribute.

2073 AttributeConsumingServiceIndex [Optional]

2074 Indirectly identifies information associated with the requester describing the SAML attributes the
2075 requester desires or requires to be supplied by the identity provider in the <Response> message. The
2076 identity provider MUST have a trusted means to map the index value in the attribute to information
2077 associated with the requester. [SAMLMeta] provides one possible mechanism. The identity provider
2078 MAY use this information to populate one or more <saml:AttributeStatement> elements in the
2079 assertion(s) it returns.

2080 ProviderName [Optional]

2081 Specifies the human-readable name of the requester for use by the presenter's user agent or the
2082 identity provider.

2083 See Section 3.4.1.4 for general processing rules regarding this message.

2084 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**
2085 complex type:

```
2086 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
2087 <complexType name="AuthnRequestType">
2088   <complexContent>
2089     <extension base="samlp:RequestAbstractType">
2090       <sequence>
2091         <element ref="saml:Subject" minOccurs="0"/>
2092         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
2093         <element ref="saml:Conditions" minOccurs="0"/>
2094         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
2095         <element ref="samlp:Scoping" minOccurs="0"/>
2096       </sequence>
2097       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2098       <attribute name="IsPassive" type="boolean" use="optional"/>
2099       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2100       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2101 use="optional"/>
2102       <attribute name="AssertionConsumerServiceURL" type="anyURI"
2103 use="optional"/>
2104       <attribute name="AttributeConsumingServiceIndex"
2105 type="unsignedShort" use="optional"/>
2106       <attribute name="ProviderName" type="string" use="optional"/>
2107     </extension>
2108   </complexContent>
2109 </complexType>
```

2110 3.4.1.1 Element <NameIDPolicy>

2111 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
2112 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

2113 Format [Optional]

2114 Specifies the URI reference corresponding to a name identifier format defined in this or another
2115 specification (see Section 8.3 for examples). The additional value of
2116 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use
2117 within this attribute to indicate a request that the resulting identifier be encrypted.

2118 SPNameQualifier [Optional]

2119 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of
2120 a service provider other than the requester, or in the namespace of an affiliation group of service
2121 providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-`
2122 `format:persistent` in Section 8.3.7.

2123 AllowCreate [Optional]

2124 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the
2125 request, to create a new identifier to represent the principal. Defaults to "false". When "false", the
2126 requester constrains the identity provider to only issue an assertion to it if an acceptable identifier for
2127 the principal has already been established. Note that this does not prevent the identity provider from
2128 creating such identifiers outside the context of this specific request (for example, in advance for a
2129 large number of principals).

2130 When this element is used, if the content is not understood by or acceptable to the identity provider, then a
2131 `<Response>` message element MUST be returned with an error `<Status>`, and MAY contain a second-
2132 level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

2133 If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-`
2134 `format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any
2135 additional constraints due to the content of this element or the policies of the identity provider or principal.

2136 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates
2137 that the resulting assertion(s) MUST contain `<EncryptedID>` elements instead of plaintext. The
2138 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the
2139 requested subject.

2140 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an
2141 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly
2142 specific to the service provider) require that an encrypted identifier be used.

2143 Note that if the requester wishes to permit the identity provider to establish a new identifier for the principal
2144 if none exists, it MUST include this element with the `AllowCreate` attribute set to "true". Otherwise,
2145 only a principal for whom the identity provider has previously established an identifier usable by the
2146 requester can be authenticated successfully. This is primarily useful in conjunction with the
2147 `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` `Format` value (see Section 8.3.7).

2148 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
2149 complex type:

```
2150     <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
2151     <complexType name="NameIDPolicyType">
2152       <attribute name="Format" type="anyURI" use="optional"/>
2153       <attribute name="SPNameQualifier" type="string" use="optional"/>
2154       <attribute name="AllowCreate" type="boolean" use="optional"/>
2155     </complexType>
```

2156 3.4.1.2 Element `<Scoping>`

2157 The `<Scoping>` element specifies the identity providers trusted by the requester to authenticate the
2158 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
2159 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
2160 elements and attribute:

2161 ProxyCount [Optional]

2162 Specifies the number of proxying indirections permissible between the identity provider that receives
2163 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of
2164 zero permits no proxying, while omitting this attribute expresses no such restriction.

2165 <IDPList> [Optional]
 2166 An advisory list of identity providers and associated information that the requester deems acceptable
 2167 to respond to the request.

2168 <RequesterID> [Zero or More]
 2169 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate
 2170 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for a
 2171 description of entity identifiers.

2172 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
 2173 <Response> message with an error <Status> and a second-level <StatusCode> of
 2174 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or
 2175 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support
 2176 any of the specified identity providers.

2177 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```

2178 <element name="Scoping" type="samlp:ScopingType"/>
2179 <complexType name="ScopingType">
2180   <sequence>
2181     <element ref="samlp:IDPList" minOccurs="0"/>
2182     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2183   </sequence>
2184   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2185 </complexType>
2186 <element name="RequesterID" type="anyURI"/>
  
```

2187 3.4.1.3 Element <IDPList>

2188 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the
 2189 presenter. Its **IDPListType** complex type defines the following elements:

2190 <IDPEntry> [One or More]
 2191 Information about a single identity provider.

2192 <GetComplete> [Optional]
 2193 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to
 2194 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML
 2195 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>
 2196 element.

2197 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```

2198 <element name="IDPList" type="samlp:IDPListType"/>
2199 <complexType name="IDPListType">
2200   <sequence>
2201     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2202     <element ref="samlp:GetComplete" minOccurs="0"/>
2203   </sequence>
2204 </complexType>
2205 <element name="GetComplete" type="anyURI"/>
  
```

2206 3.4.1.3.1 Element <IDPEntry>

2207 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the
 2208 presenter. Its **IDPEntryType** complex type defines the following attributes:

2209 ProviderID [Required]
 2210 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2211 Name [Optional]

2212 A human-readable name for the identity provider.

2213 Loc [Optional]

2214 A URI reference representing the location of a profile-specific endpoint supporting the authentication request protocol. The binding to be used must be understood from the profile of use.

2216 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
2217 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2218 <complexType name="IDPEntryType">
2219   <attribute name="ProviderID" type="anyURI" use="required"/>
2220   <attribute name="Name" type="string" use="optional"/>
2221   <attribute name="Loc" type="anyURI" use="optional"/>
2222 </complexType>
```

2223 3.4.1.4 Processing Rules

2224 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
2225 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
2226 kinds of input and output are required or prohibited. The following processing rules apply as invariant
2227 behavior across any profile of this protocol exchange. All other processing rules associated with the
2228 underlying request and response messages **MUST** also be observed.

2229 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing
2230 one or more assertions that meet the specifications defined by the request, or with a <Response>
2231 message containing a <Status> describing the error that occurred. The responder **MAY** conduct
2232 additional message exchanges with the presenter as needed to initiate or complete the authentication
2233 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
2234 the next section, this includes proxying the request by directing the presenter to another identity provider
2235 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to
2236 authenticate the presenter to the original responder, in effect using SAML as the authentication
2237 mechanism.

2238 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if
2239 prevented from providing an assertion by policies in effect at the identity provider (for example the
2240 intended subject has prohibited the identity provider from providing assertions to the relying party), then it
2241 **MUST** return a <Response> with an error <Status>, and **MAY** return a second-level <StatusCode> of
2242 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
2243 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2244 If the <saml:Subject> element in the request is present, then the resulting assertions'
2245 <saml:Subject> **MUST strongly match** the request <saml:Subject>, as described in Section 3.3.4,
2246 except that the identifier **MAY** be in a different format if specified by <NameIDPolicy>. In such a case,
2247 the identifier's physical content **MAY** be different, but it **MUST** refer to the same principal.

2248 All of the content defined specifically within <AuthnRequest> is optional, although some may be required
2249 by certain profiles. In the absence of any specific content at all, the following behavior is implied:

2250 • The assertion(s) returned **MUST** contain a <saml:Subject> element that represents the
2251 presenter. The identifier type and format are determined by the identity provider. At least one
2252 statement in at least one assertion **MUST** be a <saml:AuthnStatement> that describes the
2253 authentication performed by the responder or authentication service associated with it.

2254 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the
2255 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation
2256 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.

- 2257 • The resulting assertion(s) MUST contain a `<saml:AudienceRestriction>` element
2258 referencing the requester as an acceptable relying party. Other audiences MAY be included as
2259 deemed appropriate by the identity provider.

2260 **3.4.1.5 Proxying**

2261 If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or
2262 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
2263 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
2264 `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in
2265 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
2266 identity provider.

2267 Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the
2268 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
2269 proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>`,
2270 completing the overall message exchange. Both the proxying and authenticating identity providers MAY
2271 include constraints on proxying activity in the messages and assertions they issue, as described in
2272 previous sections and below.

2273 The requester can influence proxy behavior by including a `<Scoping>` element where the provider sets a
2274 desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by
2275 including an ordered `<IDPList>` of preferred providers.

2276 An identity provider can control secondary use of its assertions by proxying identity providers using a
2277 `<ProxyRestriction>` element in the assertions it issues.

2278 **3.4.1.5.1 Proxying Processing Rules**

2279 An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is
2280 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2281 choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

2282 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity
2283 provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of
2284 `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly
2285 authenticate the presenter.

2286 If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying
2287 identity provider MUST include equivalent or stricter forms of all the information included in the original
2288 request (such as authentication context policy). Note, however, that the proxying provider is free to specify
2289 whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

2290 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
2291 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for
2292 example) will be honored by the authenticating provider.

2293 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less
2294 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new
2295 request SHOULD contain a `<ProxyCount>` attribute.

2296 If an `<IDPList>` was specified in the original request, the new request MUST also contain an
2297 `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the
2298 `<IDPList>`, but MUST NOT remove any from the list.

2299 The authentication request and response are processed in normal fashion, in accordance with the rules
2300 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
2301 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2302 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
2303 relevant information from the original assertion or non-SAML equivalent.
- 2304 • The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original
2305 requester 's preferences, as defined by its <NameIDPolicy> element.
- 2306 • The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext>
2307 element containing a <saml:AuthenticatingAuthority> element referencing the identity
2308 provider to which the proxying identity provider referred the presenter. If the original assertion
2309 contains <saml:AuthnContext> information that includes one or more
2310 <saml:AuthenticatingAuthority> elements, those elements SHOULD be included in the
2311 new assertion, with the new element placed after them.
- 2312 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
2313 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
2314 consistent over time across different requests. The value MUST not conflict with values used or
2315 generated by other SAML providers.
- 2316 • Any other <saml:AuthnContext> information MAY be copied, translated, or omitted in
2317 accordance with the policies of the proxying identity provider, provided that the original
2318 requirements dictated by the requester are met.

2319 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,
2320 and this request is equally or less strict than the original request (as determined by the proxying identity
2321 provider), the identity provider MAY skip the creation of a new <AuthnRequest> to the authenticating
2322 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML
2323 equivalent it received is still valid).

2324 **3.5 Artifact Resolution Protocol**

2325 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
2326 transported in a SAML binding by reference instead of by value. Both requests and responses can be
2327 obtained by reference using this specialized protocol. A message sender, instead of binding a message to
2328 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2329 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver
2330 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
2331 protocol to resolve the artifact into the original protocol message.

2332 The most common use for this mechanism is with bindings that cannot easily carry a message because of
2333 size constraints, or to enable a message to be communicated via a secure channel between the SAML
2334 requester and responder, avoiding the need for a signature.

2335 Depending on the characteristics of the underlying message being passed by reference, the artifact
2336 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2337 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
2338 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
2339 by any party.

2340 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
2341 as if the message so obtained had been sent originally in place of the artifact.

2342 3.5.1 Element <ArtifactResolve>

2343 The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an
2344 <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message.
2345 The original transmission of the artifact is governed by the specific protocol binding that is being used; see
2346 [SAMLBind] for more information on the use of artifacts in bindings.

2347 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
2348 protected by the protocol binding used to deliver the message.

2349 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
2350 adds the following element:

2351 <Artifact> [Required]

2352 The artifact value that the requester received and now wishes to translate into the protocol message it
2353 represents. See [SAMLBind] for specific artifact format information.

2354 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**
2355 complex type:

```
2356 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
2357 <complexType name="ArtifactResolveType">
2358   <complexContent>
2359     <extension base="samlp:RequestAbstractType">
2360       <sequence>
2361         <element ref="samlp:Artifact"/>
2362       </sequence>
2363     </extension>
2364   </complexContent>
2365 </complexType>
2366 <element name="Artifact" type="string"/>
```

2367 3.5.2 Element <ArtifactResponse>

2368 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
2369 message element. This element is of complex type **ArtifactResponseType**, which extends
2370 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2371 message being returned. This wrapped message element can be a request or a response.

2372 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2373 protected by the protocol binding used to deliver the message.

2374 The following schema fragment defines the <ArtifactResponse> element and its
2375 **ArtifactResponseType** complex type:

```
2376 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2377 <complexType name="ArtifactResponseType">
2378   <complexContent>
2379     <extension base="samlp:StatusResponseType">
2380       <sequence>
2381         <any namespace="##any" processContents="lax" minOccurs="0"/>
2382       </sequence>
2383     </extension>
2384   </complexContent>
2385 </complexType>
```

2386 3.5.3 Processing Rules

2387 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2388 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>

2389 element with no embedded message. In both cases, the <Status> element MUST include a
2390 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A
2391 response message with no embedded message inside it is termed an empty response in the remainder of
2392 this section.

2393 The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent
2394 request with the same artifact by any requester results in an empty response as described above.

2395 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY
2396 be intended for consumption by any party that receives it and can respond appropriately. In most other
2397 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2398 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2399 receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the
2400 original intended recipient, then the artifact issuer MUST return an empty response.

2401 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2402 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2403 return it in an <ArtifactResolve> message to the issuer.

2404 Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of
2405 the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message
2406 will contain its own message identifier, and in the case of an embedded response, may contain a different
2407 InResponseTo value that corresponds to the original request message to which the embedded message
2408 is responding.

2409 All other processing rules associated with the underlying request and response messages MUST be
2410 observed.

2411 **3.6 Name Identifier Management Protocol**

2412 After establishing a name identifier for a principal, an identity provider wishing to change the value and/or
2413 format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will
2414 no longer be used to refer to the principal, informs service providers of the change by sending them a
2415 <ManageNameIDRequest> message.

2416 A service provider also uses this message to register or change the SPProvidedID value to be included
2417 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
2418 identifier between itself and the identity provider.

2419 Note that this protocol is typically not used with "transient" name identifiers, since their value is not
2420 intended to be managed on a long term basis.

2421 **3.6.1 Element <ManageNameIDRequest>**

2422 A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name
2423 identifier or to indicate the termination of the use of a name identifier.

2424 The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity
2425 protected by the protocol binding used to deliver the message.

2426 This message has the complex type **ManageNameIDRequestType**, which extends
2427 **RequestAbstractType** and adds the following elements:

2428 <saml:NameID> OR <saml:EncryptedID> [Required]

2429 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
2430 principal as currently recognized by the identity and service providers prior to this request. (For more
2431 information on these elements, see Section 2.2.)

2432 <NewID> or <NewEncryptedID> or <Terminate> [Required]

2433 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2434 requesting provider concerning this principal, or an indication that the use of the old identifier has
2435 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
2436 appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the
2437 identity provider, the new value will appear in subsequent <NameID> elements as the element's
2438 content.

2439 The following schema fragment defines the <ManageNameIDRequest> element and its
2440 **ManageNameIDRequestType** complex type:

```
2441 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2442 <complexType name="ManageNameIDRequestType">
2443   <complexContent>
2444     <extension base="samlp:RequestAbstractType">
2445       <sequence>
2446         <choice>
2447           <element ref="saml:NameID"/>
2448           <element ref="saml:EncryptedID"/>
2449         </choice>
2450         <choice>
2451           <element ref="samlp:NewID"/>
2452           <element ref="samlp:NewEncryptedID"/>
2453           <element ref="samlp:Terminate"/>
2454         </choice>
2455       </sequence>
2456     </extension>
2457   </complexContent>
2458 </complexType>
2459 <element name="NewID" type="string"/>
2460 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
2461 <element name="Terminate" type="samlp:TerminateType"/>
2462 <complexType name="TerminateType"/>
```

2463 3.6.2 Element <ManageNameIDResponse>

2464 The recipient of a <ManageNameIDRequest> message MUST respond with a
2465 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional
2466 content.

2467 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity
2468 protected by the protocol binding used to deliver the message.

2469 The following schema fragment defines the <ManageNameIDResponse> element:

```
2470 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

2471 3.6.3 Processing Rules

2472 If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize,
2473 the responding provider MUST respond with an error <Status> and MAY respond with a second-level
2474 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2475 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the
2476 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of
2477 an identity provider) it will no longer issue assertions to the service provider about the principal. The
2478 receiving provider can perform any maintenance with the knowledge that the relationship represented by
2479 the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for
2480 whom a relationship has been terminated.

2481 If the service provider requests that its identifier for the principal be changed by including a `<NewID>` (or
2482 `<NewEncryptedID>`) element, the identity provider **MUST** include the element's content as the
2483 `SPProvidedID` when subsequently communicating to the service provider regarding this principal.

2484 If the identity provider requests that its identifier for the principal be changed by including a `<NewID>` (or
2485 `<NewEncryptedID>`) element, the service provider **MUST** use the element's content as the
2486 `<saml:NameID>` element content when subsequently communicating with the identity provider regarding
2487 this principal.

2488 Note that neither, either, or both of the original and new identifier **MAY** be encrypted (using the
2489 `<EncryptedID>` and `<NewEncryptedID>` elements).

2490 In any case, the `<saml:NameID>` content in the request and its associated `SPProvidedID` attribute
2491 **MUST** contain the most recent name identifier information established between the providers for the
2492 principal.

2493 In the case of an identifier with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`
2494 `format:persistent`, the `NameQualifier` attribute **MUST** contain the unique identifier of the identity
2495 provider that created the identifier. If the identifier was established between the identity provider and an
2496 affiliation group of which the service provider is a member, then the `SPNameQualifier` attribute **MUST**
2497 contain the unique identifier of the affiliation group. Otherwise, it **MUST** contain the unique identifier of the
2498 service provider. These attributes **MAY** be omitted if they would otherwise match the value of the
2499 containing protocol message's `<Issuer>` element, but this is **NOT RECOMMENDED** due to the
2500 opportunity for confusion.

2501 Changes to these identifiers may take a potentially significant amount of time to propagate through the
2502 systems at both the requester and the responder. Implementations might wish to allow each party to
2503 accept either identifier for some period of time following the successful completion of a name identifier
2504 change. Not doing so could result in the inability of the principal to access resources.

2505 All other processing rules associated with the underlying request and response messages **MUST** be
2506 observed.

2507 **3.7 Single Logout Protocol**

2508 The single logout protocol provides a message exchange protocol by which all sessions provided by a
2509 particular session authority are near-simultaneously terminated. The single logout protocol is used either
2510 when a principal logs out at a session participant or when the principal logs out directly at the
2511 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
2512 the logout event can be indicated through the `Reason` attribute.

2513 The principal may have established authenticated sessions with both the session authority and individual
2514 session participants, based on assertions containing authentication statements supplied by the session
2515 authority.

2516 When the principal invokes the single logout process at a session participant, the session participant
2517 **MUST** send a `<LogoutRequest>` message to the session authority that provided the assertion
2518 containing the authentication statement related to that session at the session participant.

2519 When either the principal invokes a logout at the session authority, or a session participant sends a logout
2520 request to the session authority specifying that principal, the session authority **SHOULD** send a
2521 `<LogoutRequest>` message to each session participant to which it provided assertions containing
2522 authentication statements under its current session with the principal, with the exception of the session
2523 participant that sent the `<LogoutRequest>` message to the session authority. It **SHOULD** attempt to
2524 contact as many of these participants as it can using this protocol, terminate its own session with the
2525 principal, and finally return a `<LogoutResponse>` message to the requesting session participant, if any.

2529 3.7.1 Element <LogoutRequest>

2530 A session participant or session authority sends a <LogoutRequest> message to indicate that a session
2531 has been terminated.

2532 The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected
2533 by the protocol binding used to deliver the message.

2534 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and
2535 adds the following elements and attributes:

2536 NotOnOrAfter [Optional]

2537 The time at which the request expires, after which the recipient may discard the message. The time
2538 value is encoded in UTC, as described in Section 1.3.3.

2539 Reason [Optional]

2540 An indication of the reason for the logout, in the form of a URI reference.

2541 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2542 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as
2543 currently recognized by the identity and service providers prior to this request. (For more information
2544 on this element, see Section 2.2.)

2545 <SessionIndex> [Optional]

2546 The identifier that indexes this session at the message recipient.

2547 The following schema fragment defines the <LogoutRequest> element and associated
2548 **LogoutRequestType** complex type:

```
2549 <element name="LogoutRequest" type="saml:LogoutRequestType"/>
2550 <complexType name="LogoutRequestType">
2551   <complexContent>
2552     <extension base="saml:RequestAbstractType">
2553       <sequence>
2554         <choice>
2555           <element ref="saml:BaseID"/>
2556           <element ref="saml:NameID"/>
2557           <element ref="saml:EncryptedID"/>
2558         </choice>
2559         <element ref="saml:SessionIndex" minOccurs="0"
2560 maxOccurs="unbounded"/>
2561       </sequence>
2562       <attribute name="Reason" type="string" use="optional"/>
2563       <attribute name="NotOnOrAfter" type="dateTime"
2564 use="optional"/>
2565     </extension>
2566   </complexContent>
2567 </complexType>
2568 <element name="SessionIndex" type="string"/>
```

2569 3.7.2 Element <LogoutResponse>

2570 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of
2571 type **StatusResponseType**, with no additional content specified.

2572 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
2573 protected by the protocol binding used to deliver the message.

2574 The following schema fragment defines the <LogoutResponse> element:

2575 `<element name="LogoutResponse" type="samlp:StatusResponseType"/>`

2576 **3.7.3 Processing Rules**

2577 The message sender MAY use the `Reason` attribute to indicate the reason for sending the
2578 `<LogoutRequest>`. The following values are defined by this specification for use by all message
2579 senders; other values MAY be agreed on between participants:

2580 `urn:oasis:names:tc:SAML:2.0:logout:user`

2581 Specifies that the message is being sent because the principal wishes to terminate the indicated
2582 session.

2583 `urn:oasis:names:tc:SAML:2.0:logout:admin`

2584 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2585 session for that principal.

2586 All other processing rules associated with the underlying request and response messages MUST be
2587 observed.

2588 Additional processing rules are provided in the following sections.

2589 **3.7.3.1 Session Participant Rules**

2590 When a session participant receives a `<LogoutRequest>` message, the session participant MUST
2591 authenticate the message. If the sender is the authority that provided an assertion containing an
2592 authentication statement linked to the principal's current session, the session participant MUST invalidate
2593 the principal's session(s) referred to by the `<saml:BaseID>`, `<saml:NameID>`, or
2594 `<saml:EncryptedID>` element, and any `<SessionIndex>` elements supplied in the message. If no
2595 `<SessionIndex>` elements are supplied, then all sessions associated with the principal MUST be
2596 invalidated.

2597
2598 The session participant MUST apply the logout request message to any assertion that meets the following
2599 conditions, even if the assertion arrives after the logout request:

- 2600 • The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or
2601 `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in Section 3.3.4.
- 2602 • The `SessionIndex` attribute of one of the assertion's authentication statements matches one of
2603 the `<SessionIndex>` elements specified in the logout request, or the logout request contains no
2604 `<SessionIndex>` elements.
- 2605 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself
2606 (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject
2607 confirmation data).
- 2608 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on
2609 the message).

2610 **Note:** This rule is intended to prevent a situation in which a session participant receives a
2611 logout request targeted at a single, or multiple, assertion(s) (as identified by the
2612 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid –
2613 assertion(s) targeted by the logout request. It should honor the logout request until the
2614 logout request itself may be discarded (the `NotOnOrAfter` value on the request has
2615 been exceeded) or the assertion targeted by the logout request has been received and
2616 has been handled appropriately.

2617 3.7.3.2 Session Authority Rules

2618 When a session authority receives a `<LogoutRequest>` message, the session authority MUST
2619 authenticate the sender. If the sender is a session participant to which the session authority provided an
2620 assertion containing an authentication statement for the current session, then the session authority
2621 SHOULD do the following in the specified order:

- 2622 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session
2623 authority proxied the principal's authentication, unless the second authority is the originator of the
2624 `<LogoutRequest>`.
- 2625 • Send a `<LogoutRequest>` message to each session participant for which the session authority
2626 provided assertions in the current session, *other than* the originator of a current
2627 `<LogoutRequest>`.
- 2628 • Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`,
2629 or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout
2630 request message.

2631 If the session authority successfully terminates the principal's session with respect to itself, then it MUST
2632 respond to the original requester, if any, with a `<LogoutResponse>` message containing a top-level
2633 status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it MUST
2634 respond with a `<LogoutResponse>` message containing a top-level status code indicating the error.
2635 Thus, the top-level status indicates the state of the logout operation only with respect to the session
2636 authority itself.

2637 The session authority SHOULD attempt to contact each session participant using any applicable/usable
2638 protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because
2639 the original request takes place using a protocol binding that does not enable the logout to be propagated
2640 to all participants).

2641 In the event that not all session participants successfully respond to these `<LogoutRequest>` messages
2642 (or if not all participants can be contacted), then the session authority MUST include in its
2643 `<LogoutResponse>` message a second-level status code of
2644 `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session
2645 participants successfully responded with confirmation of the logout.

2646 Note that a session authority MAY initiate a logout for reasons other than having received a
2647 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 2648 • If some timeout period was agreed out-of-band with an individual session participant, the session
2649 authority MAY send a `<LogoutRequest>` to that individual participant alone.
- 2650 • An agreed global timeout period has been exceeded.
- 2651 • The principal or some other trusted entity has requested logout of the principal directly at the session
2652 authority.
- 2653 • The session authority has determined that the principal's credentials may have been compromised.

2654 When constructing a logout request message, the session authority MUST set the value of the
2655 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message,
2656 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time
2657 value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most
2658 recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout
2659 request).

2660 In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also
2661 available for use by the session authority only:

2662 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2663 Specifies that the message is being sent because of the global session timeout interval period
2664 being exceeded.

2665 urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2666 Specifies that the message is being sent because a timeout interval period agreed between a
2667 participant and the session authority has been exceeded.

2668 3.8 Name Identifier Mapping Protocol

2669 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2670 identifier for the same principal in a particular format or federation namespace, it can send a request to
2671 the identity provider using this protocol.

2672 For example, a service provider that wishes to communicate with another service provider with whom it
2673 does not share an identifier for the principal can use an identity provider that shares an identifier for the
2674 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,
2675 with which it can communicate with the second service provider.

2676 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
2677 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

2678 3.8.1 Element <NameIDMappingRequest>

2679 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2680 <NameIDMappingRequest> message. This message has the complex type
2681 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2682 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2683 The identifier and associated descriptive data that specify the principal as currently recognized by the
2684 requester and the responder. (For more information on this element, see Section 2.2.)

2685 <NameIDPolicy> [Required]

2686 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2687 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2688 binding used to deliver the message.

2689 The following schema fragment defines the <NameIDMappingRequest> element and its
2690 **NameIDMappingRequestType** complex type:

```
2691 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2692 <complexType name="NameIDMappingRequestType">
2693   <complexContent>
2694     <extension base="samlp:RequestAbstractType">
2695       <sequence>
2696         <choice>
2697           <element ref="saml:BaseID"/>
2698           <element ref="saml:NameID"/>
2699           <element ref="saml:EncryptedID"/>
2700         </choice>
2701         <element ref="samlp:NameIDPolicy"/>
2702       </sequence>
2703     </extension>
2704   </complexContent>
2705 </complexType>
```

2706 **3.8.2 Element <NameIDMappingResponse>**

2707 The recipient of a <NameIDMappingRequest> message MUST respond with a
2708 <NameIDMappingResponse> message. This message has the complex type
2709 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

2710 <saml:NameID> or <saml:EncryptedID> [Required]

2711 The identifier and associated attributes that specify the principal in the manner requested, usually in
2712 encrypted form. (For more information on this element, see Section 2.2.)

2713 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2714 binding used to deliver the message.

2715 The following schema fragment defines the <NameIDMappingResponse> element and its
2716 **NameIDMappingResponseType** complex type:

```
2717 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2718 <complexType name="NameIDMappingResponseType">
2719   <complexContent>
2720     <extension base="samlp:StatusResponseType">
2721       <choice>
2722         <element ref="saml:NameID"/>
2723         <element ref="saml:EncryptedID"/>
2724       </choice>
2725     </extension>
2726   </complexContent>
2727 </complexType>
```

2728 **3.8.3 Processing Rules**

2729 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2730 <Status> containing a second-level <StatusCode> of
2731 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2732 At the responder's discretion, the
2733 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to
2734 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2735 All other processing rules associated with the underlying request and response messages MUST be
2736 observed.

2737 4 SAML Versioning

2738 The SAML specification set is versioned in two independent ways. Each is discussed in the following
2739 sections, along with processing rules for detecting and handling version differences. Also included are
2740 guidelines on when and why specific version information is expected to change in future revisions of the
2741 specification.

2742 When version information is expressed as both a Major and Minor version, it is expressed in the form
2743 *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and
2744 only if:

2745 $(Major_B > Major_A) \text{ OR } ((Major_B = Major_A) \text{ AND } (Minor_B > Minor_A))$

2746 4.1 SAML Specification Set Version

2747 Each release of the SAML specification set will contain a major and minor version designation describing
2748 its relationship to earlier and later versions of the specification set. The version will be expressed in the
2749 content and filenames of published materials, including the specification set documents and XML schema
2750 documents. There are no normative processing rules surrounding specification set versioning, since it
2751 merely encompasses the collective release of normative specification documents which themselves
2752 contain processing rules.

2753 The overall size and scope of changes to the specification set documents will informally dictate whether a
2754 set of changes constitutes a major or minor revision. In general, if the specification set is backwards
2755 compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain
2756 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
2757 revision.

2758 4.1.1 Schema Version

2759 As a non-normative documentation mechanism, any XML schema documents published as part of the
2760 specification set will contain a `version` attribute on the `<xs:schema>` element whose value is in the
2761 form *Major.Minor*, reflecting the specification set version in which it has been published. Validating
2762 implementations MAY use the attribute as a means of distinguishing which version of a schema is being
2763 used to validate messages, or to support multiple versions of the same logical schema.

2764 4.1.2 SAML Assertion Version

2765 The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the
2766 assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed
2767 so as to document the syntax, semantics, and processing rules of the assertions of the same version.
2768 That is, specification set version 1.0 describes assertion version 1.0, and so on.

2769 There is explicitly NO relationship between the assertion version and the target XML namespace specified
2770 for the schema definitions for that assertion version.

2771 The following processing rules apply:

- 2772 • A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion
2773 version number not supported by the authority.
- 2774 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
2775 supported by the relying party.
- 2776 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version
2777 number is higher than the minor assertion version number supported by the relying party. However,
2778 all assertions that share a major assertion version number MUST share the same general

2779 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For
2780 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the
2781 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the
2782 likely effects of schema evolution.)

2783 **4.1.3 SAML Protocol Version**

2784 The various SAML protocols' request and response elements contain an attribute for expressing the major
2785 and minor version of the request or response message using a string of the form *Major.Minor*. Each
2786 version of the SAML specification set will be construed so as to document the syntax, semantics, and
2787 processing rules of the protocol messages of the same version. That is, specification set version 1.0
2788 describes request and response version V1.0, and so on.

2789 There is explicitly NO relationship between the protocol version and the target XML namespace specified
2790 for the schema definitions for that protocol version.

2791 The version numbers used in SAML protocol request and response elements will match for any particular
2792 revision of the SAML specification set.

2793 **4.1.3.1 Request Version**

2794 The following processing rules apply to requests:

- 2795 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2796 SAML requester and the SAML responder.
- 2797 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD
2798 assume that the responder supports requests with the highest request version supported by the
2799 requester.
- 2800 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version
2801 number matching a response version number that the requester does not support.
- 2802 • A SAML responder MUST reject any request with a major request version number not supported by
2803 the responder.
- 2804 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2805 higher than the highest supported request version that it supports. However, all requests that share
2806 a major request version number MUST share the same general processing rules and semantics,
2807 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the
2808 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill
2809 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

2810 **4.1.3.2 Response Version**

2811 The following processing rules apply to responses:

- 2812 • A SAML responder MUST NOT issue a response message with a response version number higher
2813 than the request version number of the corresponding request message.
- 2814 • A SAML responder MUST NOT issue a response message with a major response version number
2815 lower than the major request version number of the corresponding request message except to
2816 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2817 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
2818 top-level `<StatusCode>` value of
2819 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting
2820 one of the following second-level values:

2821 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,
2822 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or
2823 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

2824 **4.1.3.3 Permissible Version Combinations**

2825 Assertions of a particular major version appear only in response messages of the same major version, as
2826 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2827 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2828 response message, if the appropriate assertion schema is referenced during namespace importation. But
2829 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major
2830 versions.

2831 **4.2 SAML Namespace Version**

2832 XML schema documents published as part of the specification set contain one or more target
2833 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct
2834 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that
2835 part of the specification.

2836 The namespace URI references defined by the specification set will generally contain version information
2837 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond
2838 to the major and minor version of the specification set in which the namespace is first introduced and
2839 defined. This information is not typically consumed by an XML processor, which treats the namespace
2840 opaquely, but is intended to communicate the relationship between the specification set and the
2841 namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that are
2842 listed in Section 8.

2843 As a general rule, implementers can expect the namespaces and the associated schema definitions
2844 defined by a major revision of the specification set to remain valid and stable across minor revisions of the
2845 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but
2846 this is expected to be rare. In such cases, the older namespaces and their associated definitions should
2847 be expected to remain valid until a major specification set revision.

2848 **4.2.1 Schema Evolution**

2849 In general, maintaining namespace stability while adding or changing the content of a schema are
2850 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2851 older implementations will react to any given change, making forward compatibility difficult to achieve.
2852 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
2853 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),
2854 implementations should expect forward-compatible schema changes in minor revisions, allowing new
2855 messages to validate against older schemas.

2856 Implementations SHOULD expect and be prepared to deal with new extensions and message types in
2857 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types
2858 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such
2859 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
2860 include new query, statement, or condition types.

2861

5 SAML and XML Signature Syntax and Processing

2862 SAML assertions and SAML protocol request and response messages may be signed, with the following
2863 benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the
2864 asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-
2865 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the
2866 message originator supports message integrity, authentication of message origin to a destination, and, if
2867 the signature is based on the originator's public-private key pair, non-repudiation of origin.

2868 A digital signature is not always required in SAML. For example, in some circumstances, signatures may
2869 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing
2870 protocol response message. "Inherited" signatures should be used with care when the contained object
2871 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context
2872 must be retained to allow validation, exposing the XML content and adding potentially unnecessary
2873 overhead. As another example, the SAML relying party or SAML requester may have obtained an
2874 assertion or protocol message from the SAML asserting party or SAML responder directly (with no
2875 intermediaries) through a secure channel, with the asserting party or SAML responder having
2876 authenticated to the relying party or SAML responder by some means other than a digital signature.

2877 Many different techniques are available for "direct" authentication and secure channel establishment
2878 between two parties. The list includes TLS/SSL (see [RFC 2246]/[SSL3]), HMAC, password-based
2879 mechanisms, and so on. In addition, the applicable security requirements depend on the communicating
2880 applications and the nature of the assertion or message transported. It is RECOMMENDED that, in all
2881 other contexts, digital signatures be used for assertions and request and response messages.
2882 Specifically:

- 2883 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting
2884 party SHOULD be signed by the SAML asserting party.
- 2885 • A SAML protocol message arriving at a destination from an entity other than the originating sender
2886 SHOULD be signed by the sender.
- 2887 • Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that
2888 contain SAML documents. Caveats about retaining context and interoperability apply. XML
2889 Signatures are intended to be the primary SAML signature mechanism, but this specification
2890 attempts to ensure compatibility with profiles that may require other mechanisms.
- 2891 • Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be
2892 enveloped.

2893 5.1 Signing Assertions

2894 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as
2895 described in Section 2.

2896 5.2 Request/Response Signing

2897 All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected
2898 in the schema as described in Section 3.

2899 5.3 Signature Inheritance

2900 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
2901 or a request or response, which may be signed. When a SAML assertion does not contain a
2902 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
2903 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,

2904 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
2905 interpretation should be equivalent to the case where the assertion itself was signed with the same key
2906 and signature options.

2907 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
2908 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY
2909 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
2910 information from the surrounding context, but no such inheritance should be inferred unless specifically
2911 identified by the profile.

2912 **5.4 XML Signature Profile**

2913 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
2914 and many choices. This section details constraints on these facilities so that SAML processors do not
2915 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
2916 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**
2917 attribute on `<Assertion>` and the various request and response elements. These attributes are
2918 collectively referred to in this section as the identifier attributes.

2919 Note that this profile only applies to the use of the `<ds:Signature>` elements found directly within SAML
2920 assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to
2921 SAML content are free to define other approaches.

2922 **5.4.1 Signing Formats and Algorithms**

2923 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
2924 detached.

2925 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
2926 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
2927 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

2928 **5.4.2 References**

2929 SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of
2930 the assertion or protocol message being signed. The assertion's or protocol message's root element may
2931 or may not be the root element of the actual XML document containing the signed assertion or protocol
2932 message (e.g., it might be contained within a SOAP envelope).

2933 Signatures MUST contain a single `<ds:Reference>` containing a same-document reference to the **ID**
2934 attribute value of the root element of the assertion or protocol message being signed. For example, if the
2935 **ID** attribute value is "foo", then the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

2936 **5.4.3 Canonicalization Method**

2937 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
2938 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a
2939 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over
2940 SAML messages embedded in an XML context can be verified independent of that context.

2941 **5.4.4 Transforms**

2942 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
2943 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive

2944 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
2945 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2946 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
2947 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can
2948 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
2949 applying the transforms manually to the content and reverifying the result as consisting of the same SAML
2950 message.

2951 **5.4.5 KeyInfo**

2952 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of
2953 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be
2954 absent.

2955 **5.4.6 Example**

2956 Following is an example of a signed response containing a signed assertion. Line breaks have been
2957 added for readability; the signatures are not valid and cannot be successfully verified.

```
2958 <Response
2959   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
2960   ID="_c7055387-af61-4fce-8b98-e2927324b306"
2961   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
2962   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2963   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
2964   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2965     <ds:SignedInfo>
2966       <ds:CanonicalizationMethod
2967         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2968       <ds:SignatureMethod
2969         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2970       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">
2971         <ds:Transforms>
2972           <ds:Transform
2973             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
2974 signature" />
2975           <ds:Transform
2976             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2977             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
2978               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2979             </ds:Transform>
2980           </ds:Transforms>
2981           <ds:DigestMethod
2982             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2983           <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
2984           </ds:Reference>
2985         </ds:SignedInfo>
2986         <ds:SignatureValue>
2987           x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwCRKrtwPS6vdVNCcY5rHaFPYwkf+5
2988           ELYcPzx+pXlh43SmwviCgXRjRtMANWbHLhWAptaKlywS7gFgsD01qjyen3CP+m3D
2989           w6vKhaqledl0BYyriZb4KkHO4ahNyBVXbJwqv5pUaE4=
2990         </ds:SignatureValue>
2991         <ds:KeyInfo>
2992           <ds:X509Data>
2993             <ds:X509Certificate>
2994               MIICyJCCAajOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgakkxZAJBgNVBAYTA1VT
2995               MRlWEAyDVQQIEw1XaXNjb25zaW4xEDA0BgNVBACTB01hZGlzb24xIDAeBgNVBAoT
2996               F1VuaXZlcuNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLExJEaXZpc2lubiBvZiBJ
2997               bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIiwYDVQDExIRVBLSSBTZXJ2ZXIgdQ0Eg
2998               LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsxCzAJBgNVBAYTA1VTMRUwDwYDVQQLIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2999               CzAJBgNVBAYTA1VTMRUwDwYDVQQLIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
```

```

3000 Ym9yMQ4wDAYDVQKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3001 dTEncUCUGCSqGSib3DQEJARyYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWRR1MIGfMA0G
3002 CSqGSib3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
3003 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBiaOAPSZB113R6+KYiE7x4XAWIrcP+
3004 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
3005 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3006 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3007 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3008 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==
3009 </ds:X509Certificate>
3010 </ds:X509Data>
3011 </ds:KeyInfo>
3012 </ds:Signature>
3013 <Status>
3014 <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
3015 </Status>
3016 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3017 IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3018 xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3019 <Issuer>https://www.opensaml.org/IDP</Issuer>
3020 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3021 <ds:SignedInfo>
3022 <ds:CanonicalizationMethod
3023 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3024 <ds:SignatureMethod
3025 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3026 <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3027 <ds:Transforms>
3028 <ds:Transform
3029 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3030 signature" />
3031 <ds:Transform
3032 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3033 <InclusiveNamespaces
3034 PrefixList="#default saml ds xs xsi"
3035 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3036 </ds:Transform>
3037 </ds:Transforms>
3038 <ds:DigestMethod
3039 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
3040 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3041 </ds:Reference>
3042 </ds:SignedInfo>
3043 <ds:SignatureValue>
3044 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3045 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
3046 MwuL/cBUj20tBZQMFN7jQ9YB7klIz3RqVL+wNmeWI4=
3047 </ds:SignatureValue>
3048 <ds:KeyInfo>
3049 <ds:X509Data>
3050 <ds:X509Certificate>
3051 MIICyCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgAKCzAJBgNVBAYTA1VT
3052 MRlWAEYDVQQIEw1lXaXNjb25zaW4xEDA0BgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
3053 Fl1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
3054 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXXJ2ZXIgc0Eg
3055 Ls0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowYysx
3056 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbGJESMBAGAlUEBxMJQW5uIEFy
3057 Ym9yMQ4wDAYDVQKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3058 dTEncUCUGCSqGSib3DQEJARyYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWRR1MIGfMA0G
3059 CSqGSib3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
3060 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBiaOAPSZB113R6+KYiE7x4XAWIrcP+
3061 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
3062 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3063 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3064 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3065 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==

```

```
3066         </ds:X509Certificate>
3067     </ds:X509Data>
3068     </ds:KeyInfo>
3069 </ds:Signature>
3070 <Subject>
3071     <NameID
3072         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3073         scott@example.org
3074     </NameID>
3075     <SubjectConfirmation
3076         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3077 </Subject>
3078 <Conditions NotBefore="2003-04-17T00:46:02Z"
3079     NotOnOrAfter="2003-04-17T00:51:02Z">
3080     <AudienceRestriction>
3081         <Audience>http://www.opensaml.org/SP</Audience>
3082     </AudienceRestriction>
3083 </Conditions>
3084 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3085     <AuthnContext>
3086         <AuthnContextClassRef>
3087             urn:oasis:names:tc:SAML:2.0:ac:classes>Password
3088         </AuthnContextClassRef>
3089     </AuthnContext>
3090 </AuthnStatement>
3091 </Assertion>
3092 </Response>
```

3093 6 SAML and XML Encryption Syntax and Processing

3094 Encryption is used as the means to implement confidentiality. The most common motives for
3095 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for
3096 competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness
3097 of some other security mechanism. For example, a secret password or key may be encrypted.

3098 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 3099 • Communications confidentiality may be provided by mechanisms associated with a particular
3100 binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS (see
3101 [RFC 2246]/[SSL3]) or SOAP Message Security mechanisms for confidentiality.
- 3102 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`
3103 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be
3104 encrypted.
- 3105 • An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- 3106 • The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- 3107 • An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

3108 6.1 General Considerations

3109 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use
3110 of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace
3111 the plaintext information in the same location within the XML instance. The `<EncryptedData>` element's
3112 `Type` attribute SHOULD be used and, if it is present, MUST have the value
3113 `http://www.w3.org/2001/04/xmleenc#Element`.

3114 Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The
3115 SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

3116 6.2 Combining Signatures and Encryption

3117 Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and
3118 encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in
3119 the reverse order that signing and encryption were performed.

- 3120 • When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and
3121 placed within the `<Assertion>` element before the element is encrypted.
- 3122 • When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be
3123 performed first and then the signature calculated over the assertion or message containing the
3124 encrypted element.

3125 7 SAML Extensibility

3126 SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas.
3127 An example of an application that extends SAML assertions is the Liberty Protocols and Schema
3128 Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions
3129 and protocols.

3130 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can
3131 be combined with extensions to put the SAML framework to new uses.

3132 7.1 Schema Extension

3133 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
3134 elements can serve as the head element of a substitution group. However, SAML types are not defined as
3135 *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that
3136 extensions are typically defined only as types rather than elements, and are included in SAML instances
3137 by means of an `xsi:type` attribute.

3138 The following sections discuss only elements and types that have been specifically designed to support
3139 extensibility.

3140 7.1.1 Assertion Schema Extension

3141 The SAML assertion schema (see [SAML-XSD]) is designed to permit separate processing of the
3142 assertion package and the statements it contains, if the extension mechanism is used for either part.

3143 The following elements are intended specifically for use as extension points in an extension schema; their
3144 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 3145 • `<BaseID>` and **BaseIDAbstractType**
- 3146 • `<Condition>` and **ConditionAbstractType**
- 3147 • `<Statement>` and **StatementAbstractType**

- 3148 • The following constructs that are directly usable as part of SAML are particularly interesting targets for
3149 extension:
 - 3150 • `<AuthnStatement>` and **AuthnStatementType**
 - 3151 • `<AttributeStatement>` and **AttributeStatementType**
 - 3152 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
 - 3153 • `<AudienceRestriction>` and **AudienceRestrictionType**
 - 3154 • `<ProxyRestriction>` and **ProxyRestrictionType**
 - 3155 • `<OneTimeUse>` and **OneTimeUseType**

3156 7.1.2 Protocol Schema Extension

3157 The following SAML protocol elements are intended specifically for use as extension points in an
3158 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
3159 type:

- 3160 • `<Request>` and **RequestAbstractType**
- 3161 • `<SubjectQuery>` and **SubjectQueryAbstractType**

3162 The following constructs that are directly usable as part of SAML are particularly interesting targets for
3163 extension:

- 3164 • <AuthnQuery> and **AuthnQueryType**
- 3165 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**
- 3166 • <AttributeQuery> and **AttributeQueryType**
- 3167 • **StatusResponseType**

3168 7.2 Schema Wildcard Extension Points

3169 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes
3170 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension
3171 schema.

3172 7.2.1 Assertion Extension Points

3173 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- 3174 • <SubjectConfirmationData>: Uses **xs:anyType**, which allows any sub-elements and
3175 attributes.
- 3176 • <AuthnContextDecl>: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3177 • <AttributeValue>: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3178 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other
3179 namespaces with lax schema validation processing.

3180 The following constructs in the assertion schema allow arbitrary global attributes:

- 3181 • <Attribute> and **AttributeType**

3182 7.2.2 Protocol Extension Points

3183 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- 3184 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema
3185 validation processing.
- 3186 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax
3187 schema validation processing.
- 3188 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with
3189 lax schema validation processing. (It is specifically intended to carry a SAML request or response
3190 message element, however.)

3191 7.3 Identifier Extension

3192 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
3193 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
3194 However, it is always possible to define additional URI-based identifiers for these purposes. It is
3195 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should
3196 the meaning of a given URI used as such an identifier significantly change, or be used to mean two
3197 different things.

3198 8 SAML-Defined Identifiers

3199 The following sections define URI-based identifiers for common resource access actions, subject name
3200 identifier formats, and attribute name formats.

3201 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of
3202 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
3203 have one of the following stems, according to the specification set version in which they were first
3204 introduced:

```
3205 urn:oasis:names:tc:SAML:1.0:  
3206 urn:oasis:names:tc:SAML:1.1:  
3207 urn:oasis:names:tc:SAML:2.0:
```

3208 8.1 Action Namespace Identifiers

3209 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to
3210 common sets of actions to perform on resources.

3211 8.1.1 Read/Write/Execute/Delete/Control

3212 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

3213 Defined actions:

3214 `Read Write Execute Delete Control`

3215 These actions are interpreted as follows:

3216 `Read`

3217 The subject may read the resource.

3218 `Write`

3219 The subject may modify the resource.

3220 `Execute`

3221 The subject may execute the resource.

3222 `Delete`

3223 The subject may delete the resource.

3224 `Control`

3225 The subject may specify the access control policy for the resource.

3226 8.1.2 Read/Write/Execute/Delete/Control with Negation

3227 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

3228 Defined actions:

3229 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

3230 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions
3231 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
3232 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is
3233 affirmatively denied read permission.

3234 A SAML authority MUST NOT authorize both an action and its negated form.

3235 **8.1.3 Get/Head/Put/Post**

3236 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3237 Defined actions:

3238 GET HEAD PUT POST

3239 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
3240 the GET action on a resource is authorized to retrieve it.

3241 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST
3242 actions to the write permission. The correspondence is not exact however since an HTTP GET operation
3243 may cause data to be modified and a POST operation may cause modification to a resource other than
3244 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

3245 **8.1.4 UNIX File Permissions**

3246 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3247 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3248 The action string is a four-digit numeric code:

3249 *extended user group world*

3250 Where the *extended* access permission has the value

3251 +2 if sgid is set

3252 +4 if suid is set

3253 The *user group* and *world* access permissions have the value

3254 +1 if execute permission is granted

3255 +2 if write permission is granted

3256 +4 if read permission is granted

3257 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
3258 and execute; and world read.

3259 **8.2 Attribute Name Format Identifiers**

3260 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex
3261 type to refer to the classification of the attribute name for purposes of interpreting the name.

3262 **8.2.1 Unspecified**

3263 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3264 The interpretation of the attribute name is left to individual implementations.

3265 **8.2.2 URI Reference**

3266 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

3267 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML
3268 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-
3269 specific. See [SAMLProf] for attribute profiles that make use of this identifier.

3270 **8.2.3 Basic**

3271 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

3272 The class of strings acceptable as the attribute name MUST be drawn from the set of values belonging to
3273 the primitive type **xs:Name** as defined in [Schema2] Section 3.3.6. See [SAMLProf] for attribute profiles
3274 that make use of this identifier.

3275 **8.3 Name Identifier Format Identifiers**

3276 The following identifiers MAY be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
3277 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the
3278 associated processing rules, if any.

3279 **Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for
3280 SAML V2.0.

3281 **8.3.1 Unspecified**

3282 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3283 The interpretation of the content of the element is left to individual implementations.

3284 **8.3.2 Email Address**

3285 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3286 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
3287 defined in IETF RFC 2822 [RFC 2822] Section 3.4.1. An addr-spec has the form local-part@domain. Note
3288 that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded
3289 in parentheses) after it, and is not surrounded by "<" and ">".

3290 **8.3.3 X.509 Subject Name**

3291 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3292 Indicates that the content of the element is in the form specified for the contents of the
3293 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
3294 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
3295 differ from the rules given in IETF RFC 2253 [RFC 2253].

3296 **8.3.4 Windows Domain Qualified Name**

3297 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3298 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3299 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator
3300 MAY be omitted.

3301 **8.3.5 Kerberos Principal Name**

3302 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3303 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3304 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
3305 realm are described in IETF RFC 1510 [RFC 1510].

3306 **8.3.6 Entity Identifier**

3307 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3308 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3309 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service
3310 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to
3311 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make
3312 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be
3313 used in other elements and attributes whose purpose is to identify a system entity in various protocol
3314 exchanges.

3315 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3316 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3317 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

3318 **8.3.7 Persistent Identifier**

3319 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3320 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3321 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
3322 generated by identity providers MUST be constructed using pseudo-random values that have no
3323 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
3324 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
3325 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3326 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity
3327 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived
3328 from the context of the message containing the element, such as the issuer of a protocol message or an
3329 assertion containing the identifier in its subject. Note that a different system entity might later issue its own
3330 protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in
3331 this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be
3332 omitted in such a case).

3333 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service
3334 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be
3335 omitted if the element is contained in a message intended only for consumption directly by the service
3336 provider, and the value would be the unique identifier of that service provider.

3337 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most
3338 recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been
3339 established, then the attribute MUST be omitted.

3340 Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be shared
3341 in clear text with providers other than the providers that have established the shared identifier.
3342 Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and
3343 protections. Deployments without such requirements are free to use other kinds of identifiers in their
3344 SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

3345 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3346 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3347 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3348 different and does not affect the behavior of the identity provider or any processing rules specific to
3349 persistent identifiers in the protocols defined in this specification.

3350 Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of
3351 creation, but not of use. If a persistent identifier is created by a particular identity provider, the
3352 `NameQualifier` attribute value is permanently established at that time. If a service provider that receives
3353 such an identifier takes on the role of an identity provider and issues its own assertion containing that
3354 identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It
3355 might alternatively choose to create its own persistent identifier to represent the principal and link the two
3356 values. This is a deployment decision.

3357 **8.3.8 Transient Identifier**

3358 **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

3359 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3360 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3361 accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of
3362 256 characters.

3363 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3364 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
3365 accordance with the rules specified in Section 8.3.7.

3366 **8.4 Consent Identifiers**

3367 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** and
3368 **StatusResponseType** complex types to communicate whether a principal gave consent, and under what
3369 conditions, for the message.

3370 **8.4.1 Unspecified**

3371 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

3372 No claim as to principal consent is being made.

3373 **8.4.2 Obtained**

3374 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

3375 Indicates that a principal's consent has been obtained by the issuer of the message.

3376 **8.4.3 Prior**

3377 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`

3378 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to
3379 the action that initiated the message.

3380 **8.4.4 Implicit**

3381 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3382 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the
3383 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically
3384 more proximal to the action in time and presentation than prior consent, such as part of a session of
3385 activities.

3386 **8.4.5 Explicit**

3387 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3388 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the
3389 action that initiated the message.

3390 **8.4.6 Unavailable**

3391 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3392 Indicates that the issuer of the message did not obtain consent.

3393 **8.4.7 Inapplicable**

3394 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3395 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

3396 9 References

3397 The following works are cited in the body of this specification.

3398 9.1 Normative References

- 3399 **[Excl-C14N]** J. Boyer et al. *Exclusive XML Canonicalization Version 1.0*. World Wide Web
3400 Consortium, July 2002. See <http://www.w3.org/TR/xml-exc-c14n/>.
- 3401 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
3402 Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- 3403
- 3404 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium
3405 Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-2/>.
- 3406 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World
3407 Wide Web Consortium, October 2000. See <http://www.w3.org/TR/REC-xml>.
- 3408 **[XMLEnc]** D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide Web
3409 Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>. Note
3410 that this specification normatively references [XMLEnc-XSD], listed below.
- 3411 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium. See
3412 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3413 **[XMLNS]** T. Bray et al. *Namespaces in XML*. World Wide Web Consortium, January 1999.
3414 See <http://www.w3.org/TR/REC-xml-names>.
- 3415 **[XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web
3416 Consortium, February 2002. See <http://www.w3.org/TR/xmlsig-core/>. Note that
3417 this specification normatively references [XMLSig-XSD], listed below.
- 3418 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium. See
3419 [http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)
3420 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

3421 9.2 Non-Normative References

- 3422 **[LibertyProt]** J. Beatty et al. *Liberty Protocols and Schema Specification Version 1.1*. Liberty
3423 Alliance Project, January 2003. See
3424 [http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf)
3425 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3426 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF
3427 RFC 1510, September 1993. See <http://www.ietf.org/rfc/rfc1510.txt>.
- 3428 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
3429 RFC 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 3430 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
3431 See <http://www.ietf.org/rfc/rfc2246.txt>.
- 3432 **[RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*
3433 *Representation of Distinguished Names*. IETF RFC 2253, December 1997. See
3434 <http://www.ietf.org/rfc/rfc2253.txt>.
- 3435 **[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF
3436 RFC 2396, August, 1998. See <http://www.ietf.org/rfc/rfc2396.txt>.
- 3437 **[RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. See
3438 <http://www.ietf.org/rfc/rfc2822.txt>.

3439	[RFC 3075]	D. Eastlake, J. Reagle, D. Solo. <i>XML-Signature Syntax and Processing</i> . IETF RFC 3075, March 2001. See http://www.ietf.org/rfc/rfc3075.txt .
3440		
3441	[RFC 3513]	R. Hinden, S. Deering, <i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i> . IETF RFC 3513, April 2003. See http://www.ietf.org/rfc/rfc3513.txt .
3442		
3443	[SAMLAuthnCxt]	J. Kemp et al. <i>Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See http://www.oasis-open.org/committees/security/ .
3444		
3445		
3446	[SAMLBind]	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/ .
3447		
3448		
3449	[SAMLConform]	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. http://www.oasis-open.org/committees/security/ .
3450		
3451		
3452	[SAMLGloss]	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/ .
3453		
3454		
3455	[SAMLMeta]	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/ .
3456		
3457		
3458	[SAMLXSD]	S. Cantor et al. SAML protocols schema. OASIS SSTC, March 2005. Document ID saml-schema-protocol-2.0. See http://www.oasis-open.org/committees/security/ .
3459		
3460		
3461	[SAMLProf]	S. Cantor et al. <i>Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See http://www.oasis-open.org/committees/security/ .
3462		
3463		
3464	[SAMLSecure]	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/ .
3465		
3466		
3467		
3468	[SAMLTechOvw]	J. Hughes et al. SAML Technical Overview. OASIS, February 2005. Document ID sstc-saml-tech-overview-2.0-draft-03. See http://www.oasis-open.org/committees/security/ .
3469		
3470		
3471	[SAMLXSD]	S. Cantor et al., SAML assertions schema. OASIS SSTC, March 2005. Document ID saml-schema-assertion-2.0. See http://www.oasis-open.org/committees/security/ .
3472		
3473		
3474	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
3475		
3476	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. See http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
3477		
3478	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. See http://www.w3.org/TR/WD-charreq .
3479		
3480	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0: Normalization</i> . World Wide Web Consortium, February 2004. See http://www.w3.org/TR/charmod-norm/ .
3481		
3482		
3483	[XACML]	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See http://www.oasis-open.org/committees/xacml .
3484		
3485	[XML-ID]	J. Marsh et al. <i>xml:id Version 1.0</i> , World Wide Web Consortium, April 2004. See http://www.w3.org/TR/xml-id/ .
3486		

Appendix A. Acknowledgments

3488 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
3489 Committee, whose voting members at the time of publication were:

- 3490 • Conor Cahill, AOL
- 3491 • John Hughes, Atos Origin
- 3492 • Hal Lockhart, BEA Systems
- 3493 • Mike Beach, Boeing
- 3494 • Rebekah Metz, Booz Allen Hamilton
- 3495 • Rick Randall, Booz Allen Hamilton
- 3496 • Ronald Jacobson, Computer Associates
- 3497 • Gavenraj Sodhi, Computer Associates
- 3498 • Thomas Wisniewski, Entrust
- 3499 • Carolina Canales-Valenzuela, Ericsson
- 3500 • Dana Kaufman, Forum Systems
- 3501 • Irving Reid, Hewlett-Packard
- 3502 • Guy Denton, IBM
- 3503 • Heather Hinton, IBM
- 3504 • Maryann Hondo, IBM
- 3505 • Michael McIntosh, IBM
- 3506 • Anthony Nadalin, IBM
- 3507 • Nick Ragouzis, Individual
- 3508 • Scott Cantor, Internet2
- 3509 • Bob Morgan, Internet2
- 3510 • Peter Davis, Neustar
- 3511 • Jeff Hodges, Neustar
- 3512 • Frederick Hirsch, Nokia
- 3513 • Senthil Sengodan, Nokia
- 3514 • Abbie Barbir, Nortel Networks
- 3515 • Scott Kiestler, Novell
- 3516 • Cameron Morris, Novell
- 3517 • Paul Madsen, NTT
- 3518 • Steve Anderson, OpenNetwork
- 3519 • Ari Kermaier, Oracle
- 3520 • Vamsi Motukuru, Oracle
- 3521 • Darren Platt, Ping Identity
- 3522 • Prateek Mishra, Principal Identity
- 3523 • Jim Lien, RSA Security
- 3524 • John Linn, RSA Security
- 3525 • Rob Philpott, RSA Security
- 3526 • Dipak Chopra, SAP
- 3527 • Jahan Moreh, Sigaba
- 3528 • Bhavna Bhatnagar, Sun Microsystems

- 3529 • Eve Maler, Sun Microsystems
- 3530 • Ronald Monzillo, Sun Microsystems
- 3531 • Emily Xu, Sun Microsystems
- 3532 • Greg Whitehead, Trustgenix

3533

3534 The editors also would like to acknowledge the following former SSTC members for their contributions to
3535 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 3536 • Stephen Farrell, Baltimore Technologies
- 3537 • David Orchard, BEA Systems
- 3538 • Krishna Sankar, Cisco Systems
- 3539 • Zahid Ahmed, CommerceOne
- 3540 • Tim Alsop, CyberSafe Limited
- 3541 • Carlisle Adams, Entrust
- 3542 • Tim Moses, Entrust
- 3543 • Nigel Edwards, Hewlett-Packard
- 3544 • Joe Pato, Hewlett-Packard
- 3545 • Bob Blakley, IBM
- 3546 • Marlena Erdos, IBM
- 3547 • Marc Chanliau, Netegrity
- 3548 • Chris McLaren, Netegrity
- 3549 • Lynne Rosenthal, NIST
- 3550 • Mark Skall, NIST
- 3551 • Charles Knouse, Oblix
- 3552 • Simon Godik, Overxeer
- 3553 • Charles Norwood, SAIC
- 3554 • Evan Prodromou, Securant
- 3555 • Robert Griffin, RSA Security (former editor)
- 3556 • Sai Allarvarpu, Sun Microsystems
- 3557 • Gary Ellison, Sun Microsystems
- 3558 • Chris Ferris, Sun Microsystems
- 3559 • Mike Myers, Traceroute Security
- 3560 • Phillip Hallam-Baker, VeriSign (former editor)
- 3561 • James Vanderbeek, Vodafone
- 3562 • Mark O'Neill, Vordel
- 3563 • Tony Palmer, Vordel

3564

3565 Finally, the editors wish to acknowledge the following people for their contributions of material used as
3566 input to the OASIS Security Assertions Markup Language specifications:

- 3567 • Thomas Gross, IBM
- 3568 • Birgit Pfitzmann, IBM

Appendix B. Notices

3570 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3571 might be claimed to pertain to the implementation or use of the technology described in this document or
3572 the extent to which any license under such rights might or might not be available; neither does it represent
3573 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
3574 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
3575 available for publication and any assurances of licenses to be made available, or the result of an attempt
3576 made to obtain a general license or permission for the use of such proprietary rights by implementors or
3577 users of this specification, can be obtained from the OASIS Executive Director.

3578 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
3579 other proprietary rights which may cover technology that may be required to implement this specification.
3580 Please address the information to the OASIS Executive Director.

3581 **Copyright © OASIS Open 2005. All Rights Reserved.**

3582 This document and translations of it may be copied and furnished to others, and derivative works that
3583 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
3584 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
3585 this paragraph are included on all such copies and derivative works. However, this document itself may
3586 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
3587 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
3588 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
3589 into languages other than English.

3590 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3591 or assigns.

3592 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3593 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3594 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3595 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.