



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

Committee Draft 04, 15 January 2005

Document identifier:

sstc-saml-core-2.0-cd-04

Location:

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

Editors:

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

SAML V2.0 Contributors:

Conor P. Cahill, AOL
John Hughes, Atos Origin
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rebekah Metz, Booz Allen Hamilton
Rick Randall, Booz, Allen, Hamilton
Tim Alsop, CyberSafe Limited
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
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security

42 Jahan Moreh, Sigaba
43 Anne Anderson, Sun Microsystems
44 Gary Ellison, Sun Microsystems
45 Eve Maler, Sun Microsystems
46 Ron Monzillo, Sun Microsystems
47 Greg Whitehead, Trustgenix

48 **Abstract:**

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

51 **Status:**

52 This is a **Committee Draft** approved by the Security Services Technical Committee on 15
53 January 2005.

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

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

Table of Contents

64	1 Introduction.....	7
65	1.1 Notation.....	7
66	1.2 Schema Organization and Namespaces.....	8
67	1.3 Common Data Types.....	8
68	1.3.1 String Values.....	8
69	1.3.2 URI Values.....	9
70	1.3.3 Time Values.....	9
71	1.3.4 ID and ID Reference Values.....	9
72	2 SAML Assertions.....	11
73	2.1 Schema Header and Namespace Declarations.....	11
74	2.2 Name Identifiers.....	12
75	2.2.1 Element <BaseID>.....	12
76	2.2.2 Complex Type NameIDType.....	13
77	2.2.3 Element <NameID>.....	14
78	2.2.4 Element <EncryptedID>.....	14
79	2.2.5 Element <Issuer>.....	15
80	2.3 Assertions.....	15
81	2.3.1 Element <AssertionIDRef>.....	15
82	2.3.2 Element <AssertionURIRef>.....	15
83	2.3.3 Element <Assertion>.....	15
84	2.3.4 Element <EncryptedAssertion>.....	17
85	2.4 Subjects.....	17
86	2.4.1 Element <Subject>.....	18
87	2.4.1.1 Element <SubjectConfirmation>.....	18
88	2.4.1.2 Element <SubjectConfirmationData>.....	19
89	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	20
90	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
91	2.5 Conditions.....	21
92	2.5.1 Element <Conditions>.....	21
93	2.5.1.1 General Processing Rules.....	22
94	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
95	2.5.1.3 Element <Condition>.....	23
96	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	23
97	2.5.1.5 Element <OneTimeUse>.....	24
98	2.5.1.6 Element <ProxyRestriction>.....	25
99	2.6 Advice.....	26
100	2.6.1 Element <Advice>.....	26
101	2.7 Statements.....	26
102	2.7.1 Element <Statement>.....	26
103	2.7.2 Element <AuthnStatement>.....	26
104	2.7.2.1 Element <SubjectLocality>.....	28
105	2.7.2.2 Element <AuthnContext>.....	28
106	2.7.3 Element <AttributeStatement>.....	29
107	2.7.3.1 Element <Attribute>.....	29
108	2.7.3.1.1 Element <AttributeValue>.....	30
109	2.7.3.2 Element <EncryptedAttribute>.....	31
110	2.7.4 Element <AuthzDecisionStatement>.....	31
111	2.7.4.1 Simple Type DecisionType.....	33
112	2.7.4.2 Element <Action>.....	33

113	2.7.4.3 Element <Evidence>.....	34
114	3 SAML Protocols.....	35
115	3.1 Schema Header and Namespace Declarations.....	35
116	3.2 Requests and Responses.....	36
117	3.2.1 Complex Type RequestAbstractType.....	36
118	3.2.2 Complex Type StatusResponseType.....	38
119	3.2.2.1 Element <Status>.....	39
120	3.2.2.2 Element <StatusCode>.....	40
121	3.2.2.3 Element <StatusMessage>.....	42
122	3.2.2.4 Element <StatusDetail>.....	42
123	3.3 Assertion Query and Request Protocol.....	42
124	3.3.1 Element <AssertionIDRequest>.....	42
125	3.3.2 Queries.....	42
126	3.3.2.1 Element <SubjectQuery>.....	43
127	3.3.2.2 Element <AuthnQuery>.....	43
128	3.3.2.3 Element <RequestedAuthnContext>.....	44
129	3.3.2.4 Element <AttributeQuery>.....	45
130	3.3.2.5 Element <AuthzDecisionQuery>.....	46
131	3.3.3 Element <Response>.....	46
132	3.3.4 Processing Rules.....	47
133	3.4 Authentication Request Protocol.....	47
134	3.4.1 Element <AuthnRequest>.....	48
135	3.4.1.1 Element <NameIDPolicy>.....	50
136	3.4.1.2 Element <Scoping>.....	51
137	3.4.1.3 Element <IDPList>.....	52
138	3.4.1.3.1 Element <IDPEntry>.....	52
139	3.4.1.4 Processing Rules.....	53
140	3.4.1.5 Proxying.....	54
141	3.4.1.5.1 Proxying Processing Rules.....	54
142	3.5 Artifact Resolution Protocol.....	55
143	3.5.1 Element <ArtifactResolve>.....	56
144	3.5.2 Element <ArtifactResponse>.....	56
145	3.5.3 Processing Rules.....	56
146	3.6 Name Identifier Management Protocol.....	57
147	3.6.1 Element <ManageNameIDRequest>.....	57
148	3.6.2 Element <ManageNameIDResponse>.....	58
149	3.6.3 Processing Rules.....	58
150	3.7 Single Logout Protocol.....	59
151	3.7.1 Element <LogoutRequest>.....	60
152	3.7.2 Element <LogoutResponse>.....	60
153	3.7.3 Processing Rules.....	61
154	3.7.3.1 Session Participant Rules.....	61
155	3.7.3.2 Session Authority Rules.....	62
156	3.8 Name Identifier Mapping Protocol.....	63
157	3.8.1 Element <NameIDMappingRequest>.....	63
158	3.8.2 Element <NameIDMappingResponse>.....	64
159	3.8.3 Processing Rules.....	64
160	4 SAML Versioning.....	65
161	4.1 SAML Specification Set Version.....	65
162	4.1.1 Schema Version.....	65
163	4.1.2 SAML Assertion Version.....	65
164	4.1.3 SAML Protocol Version.....	66
165	4.1.3.1 Request Version.....	66

166	4.1.3.2 Response Version.....	66
167	4.1.3.3 Permissible Version Combinations.....	67
168	4.2 SAML Namespace Version.....	67
169	4.2.1 Schema Evolution.....	67
170	5 SAML and XML Signature Syntax and Processing.....	68
171	5.1 Signing Assertions.....	68
172	5.2 Request/Response Signing.....	68
173	5.3 Signature Inheritance.....	68
174	5.4 XML Signature Profile.....	69
175	5.4.1 Signing Formats and Algorithms.....	69
176	5.4.2 References.....	69
177	5.4.3 Canonicalization Method.....	69
178	5.4.4 Transforms.....	70
179	5.4.5 KeyInfo.....	70
180	5.4.6 Example.....	70
181	6 SAML and XML Encryption Syntax and Processing.....	73
182	6.1 General Considerations.....	73
183	6.2 Combining Signatures and Encyption.....	73
184	7 SAML Extensibility.....	74
185	7.1 Schema Extension.....	74
186	7.1.1 Assertion Schema Extension.....	74
187	7.1.2 Protocol Schema Extension.....	74
188	7.2 Schema Wildcard Extension Points.....	75
189	7.2.1 Assertion Extension Points.....	75
190	7.2.2 Protocol Extension Points.....	75
191	7.3 Identifier Extension.....	76
192	8 SAML-Defined Identifiers.....	77
193	8.1 Action Namespace Identifiers.....	77
194	8.1.1 Read/Write/Execute/Delete/Control.....	77
195	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	77
196	8.1.3 Get/Head/Put/Post.....	78
197	8.1.4 UNIX File Permissions.....	78
198	8.2 Attribute Name Format Identifiers.....	78
199	8.2.1 Unspecified.....	78
200	8.2.2 URI Reference.....	79
201	8.2.3 Basic.....	79
202	8.3 Name Identifier Format Identifiers.....	79
203	8.3.1 Unspecified.....	79
204	8.3.2 Email Address.....	79
205	8.3.3 X.509 Subject Name.....	79
206	8.3.4 Windows Domain Qualified Name.....	79
207	8.3.5 Kerberos Principal Name.....	80
208	8.3.6 Entity Identifier.....	80
209	8.3.7 Persistent Identifier.....	80
210	8.3.8 Transient Identifier.....	81
211	8.4 Consent Identifiers.....	81
212	8.4.1 Unspecified.....	81
213	8.4.2 Obtained.....	81
214	8.4.3 Prior.....	81
215	8.4.4 Implicit.....	82
216	8.4.5 Explicit.....	82

217	8.4.6 Unavailable.....	82
218	8.4.7 Inapplicable.....	82
219	9 References.....	83
220	9.1 Normative References.....	83
221	9.2 Non-Normative References.....	83
222	Appendix A. Acknowledgments.....	85
223	Appendix B. Notices.....	87
224		

225

1 Introduction

226
227
228
229
230

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

231
232
233
234
235
236

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

237
238
239
240

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

241

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

242

1.1 Notation

243
244
245

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

246
247
248

Listings of SAML schemas appear like this.

Example code listings appear like this.

249

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

250
251
252
253
254

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

255
256
257

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

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

260 1.2 Schema Organization and Namespaces

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

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

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

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

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

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

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

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

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

275 1.3 Common Data Types

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

278 1.3.1 String Values

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

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

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

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

294 Applications that compare data received in SAML documents to data from external sources MUST take
295 into account the normalization rules specified for XML. Text contained within elements is normalized so
296 that line endings are represented using linefeed characters (ASCII code 10_{Decimal}), as described in the XML
297 Recommendation [XML] §2.11. XML attribute values defined as strings (or types derived from strings) are
298 normalized as described in [XML] §3.3.3. All whitespace characters are replaced with blanks (ASCII code
299 32_{Decimal}).

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

303 1.3.2 URI Values

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

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

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

313 1.3.3 Time Values

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

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

318 1.3.4 ID and ID Reference Values

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

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

326 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the
327 implementation. In the case that a random or pseudorandom technique is employed, the probability of two
328 randomly chosen identifiers being identical MUST be less than or equal to 2^{-128} and SHOULD be less than

329 or equal to 2^{-160} . This requirement MAY be met by encoding a randomly chosen value between 128 and
330 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom
331 generator MUST be seeded with unique material in order to ensure the desired uniqueness properties
332 between different systems.

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

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

2 SAML Assertions

343

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

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

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

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

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

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

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

2.1 Schema Header and Namespace Declarations

370

371 The following schema fragment defines the XML namespaces and other header information for the
372 assertion schema:

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

```
390         Document identifier: sstc-saml-schema-assertion-2.0
391         Location: http://www.oasis-
392 open.org/committees/documents.php?wg_abbrev=security
393         Revision history:
394         V1.0 (November, 2002):
395             Initial Standard Schema.
396         V1.1 (September, 2003):
397             Updates within the same V1.0 namespace.
398         V2.0 CD-04 (January, 2005):
399             New assertion schema for SAML V2.0 namespace.
400     </documentation>
401 </annotation>
402 ...
403 </schema>
```

404 2.2 Name Identifiers

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

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

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

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

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

424 2.2.1 Element <BaseID>

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

428 NameQualifier [Optional]

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

431 SPNameQualifier [Optional]

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

435 The `NameQualifier` and `SPNameQualifier` attributes SHOULD be omitted unless the identifier's type
436 definition explicitly defines their use and semantics.

437 The following schema fragment defines the `<BaseID>` element and its **BaseIDAbstractType** complex
438 type:

```
439 <attributeGroup name="IDNameQualifiers">  
440   <attribute name="NameQualifier" type="string" use="optional"/>  
441   <attribute name="SPNameQualifier" type="string" use="optional"/>  
442 </attributeGroup>  
443 <element name="BaseID" type="saml:BaseIDAbstractType"/>  
444 <complexType name="BaseIDAbstractType" abstract="true">  
445   <attributeGroup ref="saml:IDNameQualifiers"/>  
446 </complexType>
```

447 2.2.2 Complex Type `NameIDType`

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

452 `NameQualifier` [Optional]

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

455 `SPNameQualifier` [Optional]

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

459 `Format` [Optional]

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

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

471 `SPProvidedID` [Optional]

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

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

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

```

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

```

490 2.2.3 Element <NameID>

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

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

```

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

```

496 2.2.4 Element <EncryptedID>

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

500 <xenc:EncryptedData> [Required]

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

506 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

```

519 <complexType name="EncryptedElementType">
520   <sequence>
521     <element ref="xenc:EncryptedData"/>
522     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
523   </sequence>
524 </complexType>
525 <element name="EncryptedID" type="saml:EncryptedElementType"/>

```

526 2.2.5 Element <Issuer>

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

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

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

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

534 2.3 Assertions

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

537 2.3.1 Element <AssertionIDRef>

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

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

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

544 2.3.2 Element <AssertionURIRef>

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

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

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

551 2.3.3 Element <Assertion>

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

554 `Version` [Required]

555 The version of this assertion. The identifier for the version of SAML defined in this specification is
556 "2.0". SAML versioning is discussed in Section 4.

557 `ID` [Required]

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

560 `IssueInstant` [Required]

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

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

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

568 <ds:Signature> [Optional]
569 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as
570 described below and in section 5.

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

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

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

579 Zero or more of the following statement elements:

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

583 <AuthnStatement>
584 An authentication statement.

585 <AuthzDecisionStatement>
586 An authorization decision statement.

587 <AttributeStatement>
588 An attribute statement.

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

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

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

601 If such a signature is used, then the <ds:Signature> element MUST be present, and a relying party
602 MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in
603 accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the
604 assertion. If it is valid, then the relying party SHOULD evaluate the signature to determine the identity and

605 appropriateness of the issuer and may continue to process the assertion in accordance with this
606 specification and as it deems appropriate (for example, evaluating conditions, advice, following profile-
607 specific rules, and so on).

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

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

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

631 2.3.4 Element `<EncryptedAssertion>`

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

635 `<xenc:EncryptedData>` [Required]

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

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

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

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

647 The following schema fragment defines the `<EncryptedAssertion>` element:

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

649 2.4 Subjects

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

651 2.4.1 Element <Subject>

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

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

656 Identifies the subject.

657 <SubjectConfirmation> [Zero or More]

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

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

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

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

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

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

685 2.4.1.1 Element <SubjectConfirmation>

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

689 Method [Required]

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

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

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

696 <SubjectConfirmationData> [Optional]

697 Additional confirmation information to be used by a specific confirmation method. For example, typical
698 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax
699 and Processing specification [XMLSig], which identifies a cryptographic key (See also Section
700 2.4.1.3). Particular confirmation methods MAY define a schema type to describe the elements,
701 attributes, or content that may appear in the <SubjectConfirmationData> element.

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

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

716 2.4.1.2 Element <SubjectConfirmationData>

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

723 **NotBefore** [Optional]

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

726 **NotOnOrAfter** [Optional]

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

729 **Recipient** [Optional]

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

733 **InResponseTo** [Optional]

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

737 **Address** [Optional]

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

744 Arbitrary attributes

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

752 Arbitrary elements

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

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

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

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

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

784 2.4.1.3 Complex Type KeyInfoConfirmationDataType

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

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

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

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

```
796 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
797   <complexContent>  
798     <restriction base="saml:SubjectConfirmationDataType">  
799       <sequence>  
800         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
801       </sequence>  
802     </restriction>  
803   </complexContent>  
804 </complexType>
```

805 2.4.1.4 Example of a Key-Confirmed `<Subject>`

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

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

822 2.5 Conditions

823 This section defines the SAML constructs that place constraints on the acceptable use of SAML
824 assertions.

825 2.5.1 Element `<Conditions>`

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

827 `NotBefore` [Optional]

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

830 `NotOnOrAfter` [Optional]

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

833 `<Condition>` [Any Number]

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

836 <AudienceRestriction> [Any Number]

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

838 <OneTimeUse> [Optional]

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

842 <ProxyRestriction> [Optional]

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

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

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

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

864 2.5.1.1 General Processing Rules

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

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

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

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

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

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

886 2.5.1.2 Attributes **NotBefore** and **NotOnOrAfter**

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

890 The **NotBefore** attribute specifies the time instant at which the validity interval begins. The
891 **NotOnOrAfter** attribute specifies the time instant at which the validity interval has ended.

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

899 If both attributes are present, the value for **NotBefore** MUST be less than (earlier than) the value for
900 **NotOnOrAfter**.

901 2.5.1.3 Element **<Condition>**

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

904 The following schema fragment defines the **<Condition>** element and its **ConditionAbstractType**
905 complex type:

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

908 2.5.1.4 Elements **<AudienceRestriction>** and **<Audience>**

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

914 **<Audience>**

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

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

920 The SAML asserting party cannot prevent a party to whom the assertion is disclosed from taking action on
921 the basis of the information provided. However, the **<AudienceRestriction>** element allows the
922 SAML asserting party to state explicitly that no warranty is provided to such a party in a machine- and

923 human-readable form. While there can be no guarantee that a court would uphold such a warranty
924 exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably
925 improved.

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

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

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

944 **2.5.1.5 Element `<OneTimeUse>`**

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

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

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

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

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

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

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

```
970 <element name="OneTimeUse" type="saml:OneTimeUseType"/>
```

```
971 <complexType name="OneTimeUseType">
972   <complexContent>
973     <extension base="saml:ConditionAbstractType"/>
974   </complexContent>
975 </complexType>
```

976 **2.5.1.6 Element <ProxyRestriction>**

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

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

982 **Count** [Optional]

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

985 <Audience> [Zero or More]

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

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

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

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

998 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>
999 condition is considered to always be valid. That is, this condition does not affect validity but is a condition
1000 on use.

1001 The following schema fragment defines the <ProxyRestriction> element and its

1002 **ProxyRestrictionType** complex type:

```
1003 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
1004 <complexType name="ProxyRestrictionType">
1005   <complexContent>
1006     <extension base="saml:ConditionAbstractType">
1007       <sequence>
1008         <element ref="saml:Audience" minOccurs="0"
1009 maxOccurs="unbounded"/>
1010       </sequence>
1011       <attribute name="Count" type="nonNegativeInteger" use="optional"/>
1012     </extension>
1013   </complexContent>
1014 </complexType>
```

1015 2.6 Advice

1016 This section defines the SAML constructs that contain additional information about an assertion that an
1017 asserting party wishes to provide to a relying party.

1018 2.6.1 Element <Advice>

1019 The <Advice> element contains any additional information that the SAML authority wishes to provide.
1020 This information MAY be ignored by applications without affecting either the semantics or the validity of
1021 the assertion.

1022 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,
1023 <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in
1024 other non-SAML namespaces.

1025 Following are some potential uses of the <Advice> element:

- 1026 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating
1027 the claims) or indirectly (by reference to the supporting assertions).
- 1028 • State a proof of the assertion claims.
- 1029 • Specify the timing and distribution points for updates to the assertion.

1030 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
1031 <element name="Advice" type="saml:AdviceType"/>  
1032 <complexType name="AdviceType">  
1033   <choice minOccurs="0" maxOccurs="unbounded">  
1034     <element ref="saml:AssertionIDRef"/>  
1035     <element ref="saml:AssertionURIRef"/>  
1036     <element ref="saml:Assertion"/>  
1037     <element ref="saml:EncryptedAssertion"/>  
1038     <any namespace="##other" processContents="lax"/>  
1039   </choice>  
1040 </complexType>
```

1041 2.7 Statements

1042 The following sections define the SAML constructs that contain statement information.

1043 2.7.1 Element <Statement>

1044 The <Statement> element is an extension point that allows other assertion-based applications to reuse
1045 the SAML assertion framework. SAML itself derives its core statements from this extension point. Its
1046 **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

1047 The following schema fragment defines the <Statement> element and its **StatementAbstractType**
1048 complex type:

```
1049 <element name="Statement" type="saml:StatementAbstractType"/>  
1050 <complexType name="StatementAbstractType" abstract="true"/>
```

1051 2.7.2 Element <AuthnStatement>

1052 The <AuthnStatement> element describes a statement by the SAML authority asserting that the
1053 assertion subject was authenticated by a particular means at a particular time. Assertions containing
1054 <AuthnStatement> elements MUST contain a <Subject> element.

1055 It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the
1056 following elements and attributes:

1057 **Note:** The `<AuthorityBinding>` element and its corresponding type were removed
1058 from `<AuthnStatement>` for V2.0 of SAML.

1059 `AuthnInstant` [Required]

1060 Specifies the time at which the authentication took place. The time value is encoded in UTC, as
1061 described in Section 1.3.3.

1062 `SessionIndex` [Optional]

1063 Specifies the index of a particular session between the principal identified by the subject and the
1064 authenticating authority.

1065 `SessionNotOnOrAfter` [Optional]

1066 Specifies a time instant at which the session between the principal identified by the subject and the
1067 SAML authority issuing this statement MUST be considered ended. The time value is encoded in
1068 UTC, as described in Section 1.3.3. There is no required relationship between this attribute and a
1069 `NotOnOrAfter` condition attribute that may be present in the assertion.

1070 `<SubjectLocality>` [Optional]

1071 Specifies the DNS domain name and IP address for the system from which the assertion subject was
1072 apparently authenticated.

1073 `<AuthnContext>` [Required]

1074 The context used by the authenticating authority up to and including the authentication event that
1075 yielded this statement. Contains an authentication context class reference, an authentication context
1076 declaration or declaration reference, or both. See the Authentication Context specification
1077 [SAMLAuthnCxt] for a full description of authentication context information.

1078 In general, any string value MAY be used as a `SessionIndex` value. However, when privacy is a
1079 consideration, care must be taken to ensure that the `SessionIndex` value does not invalidate other
1080 privacy mechanisms. Accordingly, the value SHOULD NOT be usable to correlate activity by a principal
1081 across different session participants. Two solutions that achieve this goal are provided below and are
1082 RECOMMENDED:

- 1083 • Use small positive integers (or reoccurring constants in a list) for the `SessionIndex`. The SAML
1084 authority SHOULD choose the range of values such that the cardinality of any one integer will be
1085 sufficiently high to prevent a particular principal's actions from being correlated across multiple session
1086 participants. The SAML authority SHOULD choose values for `SessionIndex` randomly from within
1087 this range (except when required to ensure unique values for subsequent statements given to the
1088 same session participant but as part of a distinct session).
- 1089 • Use the enclosing assertion's `ID` value in the `SessionIndex`.

1090 The following schema fragment defines the `<AuthnStatement>` element and its **AuthnStatementType**
1091 complex type:

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

```

```
1102         <attribute name="SessionNotOnOrAfter" type="dateTime"  
1103 use="optional"/>  
1104     </extension>  
1105     </complexContent>  
1106 </complexType>
```

1107 **2.7.2.1 Element <SubjectLocality>**

1108 The <SubjectLocality> element specifies the DNS domain name and IP address for the system from
1109 which the assertion subject was authenticated. It has the following attributes:

1110 Address [Optional]

1111 The network address of the system from which the principal identified by the subject was
1112 authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "1.2.3.4").
1113 IPv6 addresses SHOULD be represented as defined by section 2.2 of IETF RFC 3513 [RFC 3513]
1114 (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

1115 DNSName [Optional]

1116 The DNS name of the system from which the principal identified by the subject was authenticated.

1117 This element is entirely advisory, since both of these fields are quite easily "spoofed," but may be useful
1118 information in some applications.

1119 The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**
1120 complex type:

```
1121 <element name="SubjectLocality" type="saml:SubjectLocalityType"/>  
1122 <complexType name="SubjectLocalityType">  
1123     <attribute name="Address" type="string" use="optional"/>  
1124     <attribute name="DNSName" type="string" use="optional"/>  
1125 </complexType>
```

1126 **2.7.2.2 Element <AuthnContext>**

1127 The <AuthnContext> element specifies the context of an authentication event. The element can contain
1128 an authentication context class reference, an authentication context declaration or declaration reference,
1129 or both. Its complex **AuthnContextType** has the following elements:

1130 <AuthnContextClassRef> [Optional]

1131 A URI reference identifying an authentication context class that describes the authentication context
1132 declaration that follows.

1133 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

1134 Either an authentication context declaration provided by value, or a URI reference that identifies such
1135 a declaration. The URI reference MAY directly resolve into an XML document containing the
1136 referenced declaration.

1137 <AuthenticatingAuthority> [Zero or More]

1138 Zero or more unique identifiers of authentication authorities that were involved in the authentication of
1139 the principal (not including the assertion issuer, who is presumed to have been involved without being
1140 explicitly named here).

1141 See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication
1142 context information.

1143 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**
1144 complex type:

```

1145 <element name="AuthnContext" type="saml:AuthnContextType"/>
1146 <complexType name="AuthnContextType">
1147   <sequence>
1148     <choice>
1149       <sequence>
1150         <element ref="saml:AuthnContextClassRef"/>
1151         <choice minOccurs="0">
1152           <element ref="saml:AuthnContextDecl"/>
1153           <element ref="saml:AuthnContextDeclRef"/>
1154         </choice>
1155       </sequence>
1156       <choice>
1157         <element ref="saml:AuthnContextDecl"/>
1158         <element ref="saml:AuthnContextDeclRef"/>
1159       </choice>
1160     </choice>
1161     <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1162     maxOccurs="unbounded"/>
1163   </sequence>
1164 </complexType>
1165 <element name="AuthnContextClassRef" type="anyURI"/>
1166 <element name="AuthnContextDeclRef" type="anyURI"/>
1167 <element name="AuthnContextDecl" type="anyType"/>
1168 <element name="AuthenticatingAuthority" type="anyURI"/>

```

1169 2.7.3 Element <AttributeStatement>

1170 The <AttributeStatement> element describes a statement by the SAML authority asserting that the
1171 assertion subject is associated with the specified attributes. Assertions containing
1172 <AttributeStatement> elements MUST contain a <Subject> element.

1173 It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the
1174 following elements:

1175 <Attribute> or <EncryptedAttribute> [One or More]

1176 The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML
1177 attribute may be included with the <EncryptedAttribute> element.

1178 The following schema fragment defines the <AttributeStatement> element and its
1179 **AttributeStatementType** complex type:

```

1180 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1181 <complexType name="AttributeStatementType">
1182   <complexContent>
1183     <extension base="saml:StatementAbstractType">
1184       <choice maxOccurs="unbounded">
1185         <element ref="saml:Attribute"/>
1186         <element ref="saml:EncryptedAttribute"/>
1187       </choice>
1188     </extension>
1189   </complexContent>
1190 </complexType>

```

1191 2.7.3.1 Element <Attribute>

1192 The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the
1193 **AttributeType** complex type. It is used within an attribute statement to express particular attributes and
1194 values associated with an assertion subject, as described in the previous section. It is also used in an
1195 attribute query to request that the values of specific SAML attributes be returned (see section 3.3.2.4 for
1196 more information). The <Attribute> element contains the following XML attributes:

1197 Name [Required]

1198 The name of the attribute.

1199 NameFormat [Optional]

1200 A URI reference representing the classification of the attribute name for purposes of interpreting the
1201 name. See Section 8.2 for some URI references that MAY be used as the value of the NameFormat
1202 attribute and their associated descriptions and processing rules. If no NameFormat value is provided,
1203 the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section
1204 8.2.1) is in effect.

1205 FriendlyName [Optional]

1206 A string that provides a more human-readable form of the attribute's name, which may be useful in
1207 cases in which the actual Name is complex or opaque, such as an OID or a UUID. This attribute's
1208 value MUST NOT be used as a basis for formally identifying SAML attributes.

1209 Arbitrary attributes

1210 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to
1211 be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows
1212 additional fields to be added as needed to supply additional parameters to be used, for example, in an
1213 attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or
1214 XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a
1215 derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1216 `<AttributeValue>` [Any Number]

1217 Contains a value of the attribute. If an attribute contains more than one discrete value, it is
1218 RECOMMENDED that each value appear in its own `<AttributeValue>` element. If more than
1219 one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a
1220 datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have
1221 the identical datatype assigned.

1222 The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on
1223 its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the
1224 `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of
1225 values indicates that the requester is interested in any or all of the named attribute's values (see also
1226 Section 3.3.2.4).

1227 Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the
1228 semantics of specifying or omitting `<AttributeValue>` elements.

1229 The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
1230 <element name="Attribute" type="saml:AttributeType"/>
1231 <complexType name="AttributeType">
1232   <sequence>
1233     <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
1234   </sequence>
1235   <attribute name="Name" type="string" use="required"/>
1236   <attribute name="NameFormat" type="anyURI" use="optional"/>
1237   <attribute name="FriendlyName" type="string" use="optional"/>
1238   <anyAttribute namespace="##other" processContents="lax"/>
1239 </complexType>
```

1240 2.7.3.1.1 Element `<AttributeValue>`

1241 The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the
1242 **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

1243 If the data content of an `<AttributeValue>` element is of an XML Schema simple type (such as
1244 **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration
1245 in the `<AttributeValue>` element. If the attribute value contains structured data, the necessary data
1246 elements MAY be defined in an extension schema.

1247 **Note:** Specifying a datatype other than an XML Schema simple type on
1248 `<AttributeValue>` using `xsi:type` will require the presence of the extension schema
1249 that defines the datatype in order for schema processing to proceed.

1250 If a SAML attribute includes an empty value, such as the empty string, the corresponding
1251 `<AttributeValue>` element MUST be empty (generally this is serialized as `<AttributeValue/>`).
1252 This overrides the requirement in section 1.3.1 that string values in SAML content contain at least one
1253 non-whitespace character.

1254 If a SAML attribute includes a "null" value, the corresponding `<AttributeValue>` element MUST be
1255 empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "true" or "1".

1256 The following schema fragment defines the `<AttributeValue>` element:

```
1257 <element name="AttributeValue" type="anyType" nillable="true"/>
```

1258 2.7.3.2 Element `<EncryptedAttribute>`

1259 The `<EncryptedAttribute>` element represents a SAML attribute in encrypted fashion, as defined by
1260 the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAttribute>`
1261 element contains the following elements:

1262 `<xenc:EncryptedData>` [Required]

1263 The encrypted content and associated encryption details, as defined by the XML Encryption
1264 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
1265 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
1266 encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

1267 `<xenc:EncryptedKey>` [Zero or More]

1268 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
1269 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
1270 the `Recipient` attribute SHOULD be the URI identifier of a system entity with a SAML name
1271 identifier, as defined by Section 8.3.6.

1272 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through
1273 an intermediary.

1274 The following schema fragment defines the `<EncryptedAttribute>` element:

```
1275 <element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

1276 2.7.4 Element `<AuthzDecisionStatement>`

1277 **Note:** The `<AuthzDecisionStatement>` feature has been frozen as of SAML V2.0,
1278 with no future enhancements planned. Users who require additional functionality may
1279 want to consider the eXtensible Access Control Markup Language [XACML], which offers
1280 enhanced authorization decision features.

1281 The <AuthzDecisionStatement> element describes a statement by the SAML authority asserting that
1282 a request for access by the assertion subject to the specified resource has resulted in the specified
1283 authorization decision on the basis of some optionally specified evidence. Assertions containing
1284 <AuthzDecisionStatement> elements MUST contain a <Subject> element.

1285 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1286 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
1287 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
1288 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1289 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1290 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1291 dependent. When a scheme uses elements of the common syntax, it will also use the common
1292 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1293 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where
1294 the port is elided.

1295 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the
1296 URI normalized form wherever possible as follows:

- 1297 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1298 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1299 Inconsistent URI reference interpretation can also result from differences between the URI reference
1300 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1301 employed to specify an access control policy language. The following security conditions SHOULD be
1302 satisfied by the system which employs SAML assertions:

- 1303 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
1304 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
1305 part of the resource URI reference.
- 1306 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
1307 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
1308 gain access to a denied resource by creating such an equivalence.

1309 The <AuthzDecisionStatement> element is of type **AuthzDecisionStatementType**, which extends
1310 **StatementAbstractType** with the addition of the following elements and attributes:

1311 **Resource** [Required]

1312 A URI reference identifying the resource to which access authorization is sought. This attribute MAY
1313 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the
1314 current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1315 **Decision** [Required]

1316 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1317 the **DecisionType** simple type.

1318 <Action> [One or more]

1319 The set of actions authorized to be performed on the specified resource.

1320 <Evidence> [Optional]

1321 A set of assertions that the SAML authority relied on in making the decision.

1322 The following schema fragment defines the <AuthzDecisionStatement> element and its
1323 **AuthzDecisionStatementType** complex type:

```

1324 <element name="AuthzDecisionStatement"
1325 type="saml:AuthzDecisionStatementType"/>
1326 <complexType name="AuthzDecisionStatementType">
1327   <complexContent>
1328     <extension base="saml:StatementAbstractType">
1329       <sequence>
1330         <element ref="saml:Action" maxOccurs="unbounded"/>
1331         <element ref="saml:Evidence" minOccurs="0"/>
1332       </sequence>
1333       <attribute name="Resource" type="anyURI" use="required"/>
1334       <attribute name="Decision" type="saml:DecisionType" use="required"/>
1335     </extension>
1336   </complexContent>
1337 </complexType>

```

1338 2.7.4.1 Simple Type DecisionType

1339 The **DecisionType** simple type defines the possible values to be reported as the status of an
 1340 authorization decision statement.

1341 Permit

1342 The specified action is permitted.

1343 Deny

1344 The specified action is denied.

1345 Indeterminate

1346 The SAML authority cannot determine whether the specified action is permitted or denied.

1347 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to
 1348 provide an affirmative statement but where it is not able to issue a decision. Additional information as to
 1349 the reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>`
 1350 elements in the enclosing `<Response>`.

1351 The following schema fragment defines the **DecisionType** simple type:

```

1352 <simpleType name="DecisionType">
1353   <restriction base="string">
1354     <enumeration value="Permit"/>
1355     <enumeration value="Deny"/>
1356     <enumeration value="Indeterminate"/>
1357   </restriction>
1358 </simpleType>

```

1359 2.7.4.2 Element <Action>

1360 The `<Action>` element specifies an action on the specified resource for which permission is sought. Its
 1361 string-data content provides the label for an action sought to be performed on the specified resource, and
 1362 it has the following attribute:

1363 Namespace [Optional]

1364 A URI reference representing the namespace in which the name of the specified action is to be
 1365 interpreted. If this element is absent, the namespace
 1366 `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation` specified in Section 8.1.2 is in
 1367 effect.

1368 The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```

1369 <element name="Action" type="saml:ActionType"/>

```

```

1370 <complexType name="ActionType">
1371   <simpleContent>
1372     <extension base="string">
1373       <attribute name="Namespace" type="anyURI" use="required"/>
1374     </extension>
1375   </simpleContent>
1376 </complexType>

```

1377 2.7.4.3 Element <Evidence>

1378 The <Evidence> element contains one or more assertions or assertion references that the SAML
1379 authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains
1380 a mixture of one or more of the following elements:

1381 <AssertionIDRef> [Any number]

1382 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1383 <AssertionURIRef> [Any number]

1384 Specifies an assertion by means of a URI reference.

1385 <Assertion> [Any number]

1386 Specifies an assertion by value.

1387 <EncryptedAssertion> [Any number]

1388 Specifies an encrypted assertion by value.

1389 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1390 and the SAML authority making the authorization decision. For example, in the case that the SAML relying
1391 party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1392 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1393 assertion as valid either to the relying party or any other third party.

1394 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```

1395 <element name="Evidence" type="saml:EvidenceType"/>
1396 <complexType name="EvidenceType">
1397   <choice maxOccurs="unbounded">
1398     <element ref="saml:AssertionIDRef"/>
1399     <element ref="saml:AssertionURIRef"/>
1400     <element ref="saml:Assertion"/>
1401     <element ref="saml:EncryptedAssertion"/>
1402   </choice>
1403 </complexType>

```

3 SAML Protocols

1404

1405 SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML
1406 bindings specification [SAMLBind] describes specific means of transporting protocol messages using
1407 existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a
1408 number of applications of the protocols defined in this section together with additional processing rules,
1409 restrictions, and requirements that facilitate interoperability.

1410 Specific SAML request and response messages derive from common types. The requester sends an
1411 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1412 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1413



1415

Figure 1: SAML Request-Response Protocol

1416 In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the
1417 responder having received a corresponding request.

1418 The protocols defined by SAML achieve the following actions:

- 1419 • Returning one or more requested assertions. This can occur in response to either a direct request
1420 for specific assertions or a query for assertions that meet particular criteria.
- 1421 • Performing authentication on request and returning the corresponding assertion
- 1422 • Registering a name identifier or terminating a name registration on request
- 1423 • Retrieving a protocol message that has been requested by means of an artifact
- 1424 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
1425 request
- 1426 • Providing a name identifier mapping on request

1427 Throughout this section, text descriptions of elements and types in the SAML protocol namespace are not
1428 shown with the conventional namespace prefix `samlp:.` For clarity, text descriptions of elements and
1429 types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:.`

3.1 Schema Header and Namespace Declarations

1430

1431 The following schema fragment defines the XML namespaces and other header information for the
1432 protocol schema:

1433

```
1434 <schema  
1435   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1436   xmlns="http://www.w3.org/2001/XMLSchema"  
1437   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1438   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1439   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1440   elementFormDefault="unqualified"  
1441   attributeFormDefault="unqualified"  
   blockDefault="substitution"/>
```

```

1442     version="2.0">
1443     <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1444         schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
1445     <import namespace="http://www.w3.org/2000/09/xmlsig#"
1446         schemaLocation="http://www.w3.org/TR/2002/REC-xmlsig-core-
1447 20020212/xmlsig-core-schema.xsd"/>
1448     <annotation>
1449         <documentation>
1450             Document identifier: draft-sstc-saml-schema-protocol-2.0
1451             Location: http://www.oasis-
1452 open.org/committees/documents.php?wg_abbrev=security
1453             Revision history:
1454             V1.0 (November, 2002):
1455                 Initial Standard Schema.
1456             V1.1 (September, 2003):
1457                 Updates within the same V1.0 namespace.
1458             V2.0 CD-04 (January, 2005):
1459                 New protocol schema based in a SAML V2.0 namespace.
1460         </documentation>
1461     </annotation>
1462     ...
1463 </schema>

```

1464 3.2 Requests and Responses

1465 The following sections define the SAML constructs and basic requirements that underlie all of the request
1466 and response messages used in SAML protocols.

1467 3.2.1 Complex Type RequestAbstractType

1468 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1469 This type defines common attributes and elements that are associated with all SAML requests:

1470 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**
1471 for V2.0 of SAML.

1472 ID [Required]

1473 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section
1474 1.3.4 for identifier uniqueness. The values of the ID attribute in a request and the InResponseTo
1475 attribute in the corresponding response MUST match.

1476 Version [Required]

1477 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".
1478 SAML versioning is discussed in Section 4.

1479 IssueInstant [Required]

1480 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
1481 1.3.3.

1482 Destination [Optional]

1483 A URI reference indicating the address to which this request has been sent. This is useful to prevent
1484 malicious forwarding of requests to unintended recipients, a protection that is required by some
1485 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1486 location at which the message was received. If it does not, the request MUST be discarded. Some
1487 protocol bindings may require the use of this attribute (see [SAMLBind]).

1488 `Consent` [Optional]

1489 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1490 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value
1491 of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the
1492 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in
1493 effect.

1494 `<saml:Issuer>` [Optional]

1495 Identifies the entity that generated the request message. (For more information on this element, see
1496 Section 2.2.5.)

1497 `<ds:Signature>` [Optional]

1498 An XML Signature that authenticates the requester and provides message integrity, as described
1499 below and in Section 5.

1500 `<Extensions>` [Optional]

1501 This extension point contains optional protocol message extension elements that are agreed on
1502 between the communicating parties. No extension schema is required in order to make use of this
1503 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1504 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1505 SAML-defined namespace.

1506 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to
1507 authenticate itself, and message integrity may often be required. Authentication and message integrity
1508 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request
1509 MAY be signed, which provides both authentication of the requester and message integrity.

1510 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML
1511 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)
1512 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the
1513 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the
1514 signature to determine the identity and appropriateness of the signer and may continue to process the
1515 request or respond with an error (if the request is invalid for some other reason).

1516 If a `Consent` attribute is included and the value indicates that some form of principal consent has been
1517 obtained, then the request SHOULD be signed.

1518 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if
1519 it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value
1520 `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a
1521 suspected denial-of-service attack, not responding at all may be warranted.

1522 The following schema fragment defines the **RequestAbstractType** complex type:

```
1523 <complexType name="RequestAbstractType" abstract="true">
1524   <sequence>
1525     <element ref="saml:Issuer" minOccurs="0"/>
1526     <element ref="ds:Signature" minOccurs="0"/>
1527     <element ref="samlp:Extensions" minOccurs="0"/>
1528   </sequence>
1529   <attribute name="ID" type="ID" use="required"/>
1530   <attribute name="Version" type="string" use="required"/>
1531   <attribute name="IssueInstant" type="dateTime" use="required"/>
1532   <attribute name="Destination" type="anyURI" use="optional"/>
1533   <attribute name="Consent" type="anyURI" use="optional"/>
1534 </complexType>
1535 <element name="Extensions" type="samlp:ExtensionsType"/>
1536 <complexType name="ExtensionsType">
1537   <sequence>
```

1538
1539
1540

```
<any namespace="##other" processContents="lax" maxOccurs="unbounded"/>  
</sequence>  
</complexType>
```

1541 3.2.2 Complex Type StatusResponseType

1542 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
1543 defines common attributes and elements that are associated with all SAML responses:

1544 ID [Required]

1545 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in
1546 Section 1.3.4 for identifier uniqueness.

1547 InResponseTo [Optional]

1548 A reference to the identifier of the request to which the response corresponds, if any. If the response
1549 is not generated in response to a request, or if the ID attribute value of a request cannot be
1550 determined (for example, the request is malformed), then this attribute MUST NOT be present.
1551 Otherwise, it MUST be present and its value MUST match the value of the corresponding request's
1552 ID attribute.

1553 Version [Required]

1554 The version of this response. The identifier for the version of SAML defined in this specification is
1555 "2.0". SAML versioning is discussed in Section 4.

1556 IssueInstant [Required]

1557 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
1558 1.3.3.

1559 Destination [Optional]

1560 A URI reference indicating the address to which this response has been sent. This is useful to prevent
1561 malicious forwarding of responses to unintended recipients, a protection that is required by some
1562 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1563 location at which the message was received. If it does not, the response MUST be discarded. Some
1564 protocol bindings may require the use of this attribute (see [SAMLBind]).

1565 Consent [Optional]

1566 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1567 the sending of this response. See Section 8.4 for some URI references that MAY be used as the value
1568 of the Consent attribute and their associated descriptions. If no Consent value is provided, the
1569 identifier urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in
1570 effect.

1571 <saml:Issuer> [Optional]

1572 Identifies the entity that generated the response message. (For more information on this element, see
1573 Section 2.2.5.)

1574 <ds:Signature> [Optional]

1575 An XML Signature that authenticates the responder and provides message integrity, as described
1576 below and in Section 5.

1577 <Extensions> [Optional]

1578 This extension point contains optional protocol message extension elements that are agreed on
1579 between the communicating parties. . No extension schema is required in order to make use of this
1580 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1581 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-

1582 SAML-defined namespace.

1583 <Status> [Required]

1584 A code representing the status of the corresponding request.

1585 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to
1586 authenticate itself, and message integrity may often be required. Authentication and message integrity
1587 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML
1588 response MAY be signed, which provides both authentication of the responder and message integrity.

1589 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1590 requester receiving the response MUST verify that the signature is valid (that is, that the message has not
1591 been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on
1592 the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD
1593 evaluate the signature to determine the identity and appropriateness of the signer and may continue to
1594 process the response as it deems appropriate.

1595 If a `Consent` attribute is included and the value indicates that some form of principal consent has been
1596 obtained, then the response SHOULD be signed.

1597 The following schema fragment defines the **StatusResponseType** complex type:

```
1598 <complexType name="StatusResponseType">  
1599   <sequence>  
1600     <element ref="saml:Issuer" minOccurs="0"/>  
1601     <element ref="ds:Signature" minOccurs="0"/>  
1602     <element ref="samlp:Extensions" minOccurs="0"/>  
1603     <element ref="samlp:Status"/>  
1604   </sequence>  
1605   <attribute name="ID" type="ID" use="required"/>  
1606   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1607   <attribute name="Version" type="string" use="required"/>  
1608   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1609   <attribute name="Destination" type="anyURI" use="optional"/>  
1610   <attribute name="Consent" type="anyURI" use="optional"/>  
1611 </complexType>
```

1612 3.2.2.1 Element <Status>

1613 The <Status> element contains the following elements:

1614 <StatusCode> [Required]

1615 A code representing the status of the activity carried out in response to the corresponding request.

1616 <StatusMessage> [Optional]

1617 A message which MAY be returned to an operator.

1618 <StatusDetail> [Optional]

1619 Additional information concerning the status of the request.

1620 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1621 <element name="Status" type="samlp:StatusType"/>  
1622 <complexType name="StatusType">  
1623   <sequence>  
1624     <element ref="samlp:StatusCode"/>  
1625     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1626     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1627   </sequence>  
1628 </complexType>
```

1629 **3.2.2.2 Element <StatusCode>**

1630 The <StatusCode> element specifies a code or a set of nested codes representing the status of the
1631 corresponding request. The <StatusCode> element has the following element and attribute:

1632 Value [Required]

1633 The status code value. This attribute contains a URI reference. The value of the topmost
1634 <StatusCode> element MUST be from the top-level list provided in this section.

1635 <StatusCode> [Optional]

1636 A subordinate status code that provides more specific information on an error condition. Note that
1637 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for
1638 additional information by intentionally presenting erroneous requests.

1639 The permissible top-level <StatusCode> values are as follows:

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

1641 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1642 <StatusDetail> elements.

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

1644 The request could not be performed due to an error on the part of the requester.

1645 urn:oasis:names:tc:SAML:2.0:status:Responder

1646 The request could not be performed due to an error on the part of the SAML responder or SAML
1647 authority.

1648 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1649 The SAML responder could not process the request because the version of the request message was
1650 incorrect.

1651 The following second-level status codes are referenced at various places in this specification. Additional
1652 second-level status codes MAY be defined in future versions of the SAML specification. System entities
1653 are free to define more specific status codes by defining appropriate URI references.

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

1655 The responding provider was unable to successfully authenticate the principal.

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

1657 Unexpected or invalid content was encountered within a <saml:Attribute> or
1658 <saml:AttributeValue> element.

1659 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1660 The responding provider cannot or will not support the requested name identifier policy.

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

1662 The specified authentication context requirements cannot be met by the responder.

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

1664 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an
1665 <IDPList> can be resolved or that none of the supported identity providers are available.

1666 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1667 Indicates the responding provider cannot authenticate the principal passively, as has been requested.

1668 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
1669 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1670 supported by the intermediary.

1671 urn:oasis:names:tc:SAML:2.0:status:PartialLogout
1672 Used by a session authority to indicate to a session participant that it was not able to propagate logout
1673 to all other session participants.

1674 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
1675 Indicates that a responding provider cannot authenticate the principal directly and is not permitted to
1676 proxy the request further.

1677 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
1678 The SAML responder or SAML authority is able to process the request but has chosen not to respond.
1679 This status code MAY be used when there is concern about the security context of the request
1680 message or the sequence of request messages received from a particular requester.

1681 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
1682 The SAML responder or SAML authority does not support the request.

1683 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
1684 The SAML responder cannot process any requests with the protocol version specified in the request.

1685 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
1686 The SAML responder cannot process the request because the protocol version specified in the
1687 request message is a major upgrade from the highest protocol version supported by the responder.

1688 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
1689 The SAML responder cannot process the request because the protocol version specified in the
1690 request message is too low.

1691 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
1692 The resource value provided in the request message is invalid or unrecognized.

1693 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
1694 The response message would contain more elements than the SAML responder is able to return.

1695 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1696 An entity that has no knowledge of a particular attribute profile has been presented with an attribute
1697 drawn from that profile.

1698 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
1699 The responding provider does not recognize the principal specified or implied by the request.

1700 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
1701 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
1702 request.

1703 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
1704 type:

```
1705 <element name="StatusCode" type="samlp:StatusCodeType"/>  
1706 <complexType name="StatusCodeType">  
1707   <sequence>  
1708     <element ref="samlp:StatusCode" minOccurs="0"/>  
1709   </sequence>
```

```
1710     <attribute name="Value" type="anyURI" use="required"/>
1711 </complexType>
```

1712 3.2.2.3 Element <StatusMessage>

1713 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1714 The following schema fragment defines the <StatusMessage> element:

```
1715 <element name="StatusMessage" type="string"/>
```

1716 3.2.2.4 Element <StatusDetail>

1717 The <StatusDetail> element MAY be used to specify additional information concerning the status of
1718 the request. The additional information consists of zero or more elements from any namespace, with no
1719 requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

1720 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1721 complex type:

```
1722 <element name="StatusDetail" type="saml:StatusDetailType"/>
1723 <complexType name="StatusDetailType">
1724   <sequence>
1725     <any namespace="##any" processContents="lax" minOccurs="0"
1726     maxOccurs="unbounded"/>
1727   </sequence>
1728 </complexType>
```

1729 3.3 Assertion Query and Request Protocol

1730 This section defines messages and processing rules for requesting existing assertions by reference or
1731 querying for assertions by subject and statement type.

1732 3.3.1 Element <AssertionIDRequest>

1733 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>
1734 message element can be used to request that they be returned in a <Response> message. The
1735 <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for
1736 more information on this element.

1737 The following schema fragment defines the <AssertionIDRequest> element:

```
1738 <element name="AssertionIDRequest" type="saml:AssertionIDRequestType"/>
1739 <complexType name="AssertionIDRequestType">
1740   <complexContent>
1741     <extension base="saml:RequestAbstractType">
1742       <sequence>
1743         <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>
1744       </sequence>
1745     </extension>
1746   </complexContent>
1747 </complexType>
```

1748 3.3.2 Queries

1749 The following sections define the SAML query request messages.

1750 3.3.2.1 Element <SubjectQuery>

1751 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1752 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and
1753 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the
1754 <saml:Subject> element (defined in Section 2.4) to **RequestAbstractType**.

1755 The following schema fragment defines the <SubjectQuery> element and its
1756 **SubjectQueryAbstractType** complex type:

```
1757 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1758 <complexType name="SubjectQueryAbstractType" abstract="true">  
1759   <complexContent>  
1760     <extension base="samlp:RequestAbstractType">  
1761       <sequence>  
1762         <element ref="saml:Subject"/>  
1763       </sequence>  
1764     </extension>  
1765   </complexContent>  
1766 </complexType>
```

1767 3.3.2.2 Element <AuthnQuery>

1768 The <AuthnQuery> message element is used to make the query "What assertions containing
1769 authentication statements are available for this subject?" A successful <Response> will contain one or
1770 more assertions containing authentication statements.

1771 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1772 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1773 that have occurred in a previous interaction between the indicated subject and the authentication authority.

1774 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1775 the following element and attribute:

1776 **SessionIndex** [Optional]

1777 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1778 containing authentication statements do you have for this subject within the context of the supplied
1779 session information?"

1780 <RequestedAuthnContext> [Optional]

1781 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1782 containing authentication statements do you have for this subject that satisfy the authentication
1783 context requirements in this element?"

1784 In response to an authentication query, a SAML authority returns assertions with authentication
1785 statements as follows:

- 1786 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1787 assertions that may be returned.
- 1788 • If the **SessionIndex** attribute is present in the query, at least one <AuthnStatement> element in
1789 the set of returned assertions MUST contain a **SessionIndex** attribute that matches the
1790 **SessionIndex** attribute in the query. It is OPTIONAL for the complete set of all such matching
1791 assertions to be returned in the response.
- 1792 • If the <RequestedAuthnContext> element is present in the query, at least one
1793 <AuthnStatement> element in the set of returned assertions MUST contain an
1794 <AuthnContext> element that satisfies the element in the query (see Section 3.3.2.3). It is
1795 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1796 The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex
1797 type:

```
1798 <element name="AuthnQuery" type="samlp:AuthnQueryType"/>
1799 <complexType name="AuthnQueryType">
1800   <complexContent>
1801     <extension base="samlp:SubjectQueryAbstractType">
1802       <sequence>
1803         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
1804       </sequence>
1805       <attribute name="SessionIndex" type="string" use="optional"/>
1806     </extension>
1807   </complexContent>
1808 </complexType>
```

1809 3.3.2.3 Element <RequestedAuthnContext>

1810 The <RequestedAuthnContext> element specifies the authentication context requirements of
1811 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
1812 complex type defines the following elements and attributes:

1813 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1814 Specifies one or more URI references identifying authentication context classes or declarations.
1815 These elements are defined in Section 2.7.2.2. For more information about authentication context
1816 classes, see [SAMLAuthnCxt].

1817 Comparison [Optional]

1818 Specifies the comparison method used to evaluate the requested context classes or statements, one
1819 of "exact", "minimum", "maximum", or "better". The default is "exact".

1820 Either a set of class references or a set of declaration references can be used. The set of supplied
1821 references **MUST** be evaluated as an ordered set, where the first element is the most preferred
1822 authentication context class or declaration. If none of the specified classes or declarations can be satisfied
1823 in accordance with the rules below, then the responder **MUST** return a <Response> message with a
1824 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1825 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
1826 statement **MUST** be the exact match of at least one of the authentication contexts specified.

1827 If Comparison is set to "minimum", then the resulting authentication context in the authentication
1828 statement **MUST** be at least as strong (as deemed by the responder) as one of the authentication
1829 contexts specified.

1830 If Comparison is set to "better", then the resulting authentication context in the authentication
1831 statement **MUST** be stronger (as deemed by the responder) than any one of the authentication contexts
1832 specified.

1833 If Comparison is set to "maximum", then the resulting authentication context in the authentication
1834 statement **MUST** be as strong as possible (as deemed by the responder) without exceeding the strength
1835 of at least one of the authentication contexts specified.

1836 The following schema fragment defines the <RequestedAuthnContext> element and its
1837 **RequestedAuthnContextType** complex type:

```
1838 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1839 <complexType name="RequestedAuthnContextType">
1840   <choice>
1841     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1842     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1843   </choice>
```

```

1844     <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1845 use="optional"/>
1846 </complexType>
1847 <simpleType name="AuthnContextComparisonType">
1848   <restriction base="string">
1849     <enumeration value="exact"/>
1850     <enumeration value="minimum"/>
1851     <enumeration value="maximum"/>
1852     <enumeration value="better"/>
1853   </restriction>
1854 </simpleType>

```

1855 3.3.2.4 Element <AttributeQuery>

1856 The <AttributeQuery> element is used to make the query "Return the requested attributes for this
1857 subject." A successful response will be in the form of assertions containing attribute statements, to the
1858 extent allowed by policy. This element is of type **AttributeQueryType**, which extends
1859 **SubjectQueryAbstractType** with the addition of the following element:

1860 <saml:Attribute> [Any Number]

1861 Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no
1862 attributes are specified, it indicates that all attributes allowed by policy are requested. If a given
1863 <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if
1864 that attribute is returned in the response, it MUST NOT contain any values that are not equal to the
1865 values specified in the query. In the absence of equality rules specified by particular profiles or
1866 attributes, equality is defined as an identical XML representation of the value. For more information on
1867 <saml:Attribute>, see Section 2.7.3.1.

1868 A single query MUST NOT contain two <saml:Attribute> elements with the same Name and
1869 NameFormat values (that is, a given attribute MUST be named only once in a query).

1870 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1871 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1872 assertions that may be returned.
- 1873 • If any <Attribute> elements are present in the query, they constrain/filter the attributes and
1874 optionally the values returned, as noted above.
- 1875 • The attributes and values returned MAY also be constrained by application-specific policy
1876 considerations.

1877 The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1878 and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to
1879 indicate problems with the interpretation of attribute or value information in a query.

1880 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1881 complex type:

```

1882 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1883 <complexType name="AttributeQueryType">
1884   <complexContent>
1885     <extension base="samlp:SubjectQueryAbstractType">
1886       <sequence>
1887         <element ref="saml:Attribute" minOccurs="0"
1888 maxOccurs="unbounded"/>
1889       </sequence>
1890     </extension>
1891   </complexContent>
1892 </complexType>

```

1893 3.3.2.5 Element <AuthzDecisionQuery>

1894 The <AuthzDecisionQuery> element is used to make the query "Should these actions on this resource
1895 be allowed for this subject, given this evidence?" A successful response will be in the form of assertions
1896 containing authorization decision statements.

1897 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no
1898 future enhancements planned. Users who require additional functionality may want to
1899 consider the eXtensible Access Control Markup Language [XACML], which offers
1900 enhanced authorization decision features.

1901 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1902 addition of the following elements and attribute:

1903 Resource [Required]

1904 A URI reference indicating the resource for which authorization is requested.

1905 <saml:Action> [One or More]

1906 The actions for which authorization is requested. For more information on this element, see Section
1907 2.7.4.2.

1908 <saml:Evidence> [Optional]

1909 A set of assertions that the SAML authority MAY rely on in making its authorization decision. For more
1910 information on this element, see Section 2.7.4.3.

1911 In response to an authorization decision query, a SAML authority returns assertions with authorization
1912 decision statements as follows:

- 1913 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1914 assertions that may be returned.

1915 The following schema fragment defines the <AuthzDecisionQuery> element and its
1916 **AuthzDecisionQueryType** complex type:

```
1917 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>  
1918 <complexType name="AuthzDecisionQueryType">  
1919   <complexContent>  
1920     <extension base="samlp:SubjectQueryAbstractType">  
1921       <sequence>  
1922         <element ref="saml:Action" maxOccurs="unbounded"/>  
1923         <element ref="saml:Evidence" minOccurs="0"/>  
1924       </sequence>  
1925       <attribute name="Resource" type="anyURI" use="required"/>  
1926     </extension>  
1927   </complexContent>  
1928 </complexType>
```

1929 3.3.3 Element <Response>

1930 The <Response> message element is used when a response consists of a list of zero or more assertions
1931 that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType**
1932 and adds the following elements:

1933 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1934 Specifies an assertion by value, or optionally an encrypted assertion by value. See Section 2.3.3 for
1935 more information on these elements.

1936 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1937 <element name="Response" type="samlp:ResponseType"/>
1938 <complexType name="ResponseType">
1939   <complexContent>
1940     <extension base="samlp:StatusResponseType">
1941       <choice minOccurs="0" maxOccurs="unbounded">
1942         <element ref="saml:Assertion"/>
1943         <element ref="saml:EncryptedAssertion"/>
1944       </choice>
1945     </extension>
1946   </complexContent>
1947 </complexType>

```

1948 3.3.4 Processing Rules

1949 In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST**
 1950 contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the
 1951 query.

1952 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both
 1953 apply:

- 1954 • If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST**
 1955 include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2.
 1956 In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical"
 1957 means that the identifier element's content and attribute values **MUST** be the same. An encrypted
 1958 identifier will be identical to the original according to this definition, once decrypted.
- 1959 • If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at
 1960 least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner
 1961 described by at least one `<saml:SubjectConfirmation>` element in S2.

1962 As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular
 1963 Format value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting
 1964 S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with
 1965 different Format values and element content, even if the two identifiers are considered to refer to the
 1966 same principal.

1967 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
 1968 expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an
 1969 `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value
 1970 `urn:oasis:names:tc:SAML:2.0:status:Success`.

1971 All other processing rules associated with the underlying request and response messages **MUST** be
 1972 observed.

1973 3.4 Authentication Request Protocol

1974 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
 1975 authentication statements to establish a security context at one or more relying parties, it can use the
 1976 authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and
 1977 request that it return a `<Response>` message containing one or more such assertions. Such assertions
 1978 **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one
 1979 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

1980 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
 1981 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
 1982 is not specified, though the means of authentication might impact the content of the response. Other
 1983 issues related to the validation of authentication credentials by the identity provider or any communication

1984 between the identity provider and any other entities involved in the authentication process are also out of
1985 scope of this protocol.

1986 The descriptions and processing rules in the following sections reference the following actors, many of
1987 whom might be the same entity in a particular profile of use:

1988 Requester

1989 The entity who creates the authentication request and to whom the response is to be returned.

1990 Presenter

1991 The entity who presents the request to the identity provider and either authenticates itself during
1992 the transmission of the message, or relies on an existing security context to establish its identity. If
1993 not the requester, the presenter acts as an intermediary between the requester and the
1994 responding identity provider.

1995 Requested Subject

1996 The entity about whom one or more assertions are being requested.

1997 Attesting Entity

1998 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`
1999 elements of the resulting assertion(s).

2000 Relying Party

2001 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
2002 the profile or context of use, generally to establish a security context.

2003 Identity Provider

2004 The entity to whom the presenter gives the request and from whom the presenter receives the
2005 response.

2006 **3.4.1 Element `<AuthnRequest>`**

2007 To request that an identity provider issue an assertion with an authentication statement, a presenter
2008 authenticates to that identity provider (or relies on an existing security context) and sends it an
2009 `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to
2010 satisfy its purpose. Among these properties may be information that relates to the content of the assertion
2011 and/or information that relates to how the resulting `<Response>` message should be delivered to the
2012 requester. The process of authentication of the presenter may take place before, during, or after the initial
2013 delivery of the `<AuthnRequest>` message.

2014 The requester might not be the same as the presenter of the request if, for example, the requester is a
2015 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject
2016 so that the relying party can decide whether to provide a service.

2017 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2018 by the protocol binding used to deliver the message.

2019 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
2020 adds the following elements and attributes, all of which are optional in general, but may be required by
2021 specific profiles:

2022 `<saml:Subject>` [Optional]

2023 Specifies the requested subject of the resulting assertion(s). This may include one or more
2024 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions
2025 can be confirmed. For more information on this element, see Section 2.4.

2026 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
2027 requested subject. If no <saml:SubjectConfirmation> elements are included, then the presenter
2028 is presumed to be the only attesting entity required and the method is implied by the profile of use
2029 and/or the policies of the identity provider.

2030 <NameIDPolicy> [Optional]

2031 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
2032 then any type of identifier supported by the identity provider for the requested subject can be used,
2033 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2034 <saml:Conditions> [Optional]

2035 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting
2036 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
2037 information in this element is used as input to the process of constructing the assertion, rather than as
2038 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2039 <RequestedAuthnContext> [Optional]

2040 Specifies the requirements, if any, that the requester places on the authentication context that applies
2041 to the responding provider's authentication of the presenter. See Section 3.3.2.3 for processing rules
2042 regarding this element.

2043 <Scoping> [Optional]

2044 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as
2045 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
2046 providers by the responder.

2047 ForceAuthn [Optional]

2048 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
2049 rely on a previous security context. If a value is not provided, the default is "false". However, if both
2050 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
2051 presenter unless the constraints of IsPassive can be met.

2052 IsPassive [Optional]

2053 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take control
2054 of the user interface from the requester and interact with the presenter in a noticeable fashion. If a
2055 value is not provided, the default is "false".

2056 AssertionConsumerServiceIndex [Optional]

2057 Indirectly identifies the location to which the <Response> message should be returned to the
2058 requester. It applies only to profiles in which the requester is different from the presenter, such as the
2059 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map
2060 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one
2061 possible mechanism. If omitted, then the identity provider MUST return the <Response> message to
2062 the default location associated with the requester for the profile of use. If the index specified is invalid,
2063 then the identity provider MAY return an error <Response> or it MAY use the default location. This
2064 attribute is mutually exclusive with the AssertionConsumerServiceURL and ProtocolBinding
2065 attributes.

2066 AssertionConsumerServiceURL [Optional]

2067 Specifies by value the location to which the <Response> message MUST be returned to the
2068 requester. The responder MUST ensure by some means that the value specified is in fact associated
2069 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing
2070 <AuthnRequest> message is another. This attribute is mutually exclusive with the
2071 AssertionConsumerServiceIndex attribute and is typically accompanied by the
2072 ProtocolBinding attribute.

2073 ProtocolBinding [Optional]

2074 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>
2075 message. See [SAMLBind] for more information about protocol bindings and URI references defined
2076 for them. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute
2077 and is typically accompanied by the AssertionConsumerServiceURL attribute.

2078 AttributeConsumingServiceIndex [Optional]

2079 Indirectly identifies information associated with the requester describing the SAML attributes the
2080 requester desires or requires to be supplied by the identity provider in the <Response> message. The
2081 identity provider MUST have a trusted means to map the index value in the attribute to information
2082 associated with the requester. [SAMLMeta] provides one possible mechanism. The identity provider
2083 MAY use this information to populate one or more <saml:AttributeStatement> elements in the
2084 assertion(s) it returns.

2085 ProviderName [Optional]

2086 Specifies the human-readable name of the requester for use by the presenter's user agent or the
2087 identity provider.

2088 See Section 3.4.1.4 for general processing rules regarding this message.

2089 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**
2090 complex type:

```
2091 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
2092 <complexType name="AuthnRequestType">
2093   <complexContent>
2094     <extension base="samlp:RequestAbstractType">
2095       <sequence>
2096         <element ref="saml:Subject" minOccurs="0"/>
2097         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
2098         <element ref="saml:Conditions" minOccurs="0"/>
2099         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
2100         <element ref="samlp:Scoping" minOccurs="0"/>
2101       </sequence>
2102       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2103       <attribute name="IsPassive" type="boolean" use="optional"/>
2104       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2105       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2106 use="optional"/>
2107       <attribute name="AssertionConsumerServiceURL" type="anyURI"
2108 use="optional"/>
2109       <attribute name="AttributeConsumingServiceIndex"
2110 type="unsignedShort" use="optional"/>
2111       <attribute name="ProviderName" type="string" use="optional"/>
2112     </extension>
2113   </complexContent>
2114 </complexType>
```

2115 3.4.1.1 Element <NameIDPolicy>

2116 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
2117 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

2118 Format [Optional]

2119 Specifies the URI reference corresponding to a name identifier format defined in this or another
2120 specification (see Section 8.3 for examples). The additional value of
2121 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use
2122 within this attribute to indicate a request that the resulting identifier be encrypted.

2123 SPNameQualifier [Optional]

2124 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of
2125 a service provider other than the requester, or in the namespace of an affiliation group of service
2126 providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-`
2127 `format:persistent` in section 8.3.7.

2128 AllowCreate [Optional]

2129 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the
2130 request, to create a new identifier to represent the principal. Defaults to "false". When "false", the
2131 requester constrains the identity provider to only issue an assertion to it if an acceptable identifier for
2132 the principal has already been established. Note that this does not prevent the identity provider from
2133 creating such identifiers outside the context of this specific request (for example, in advance for a
2134 large number of principals).

2135 When this element is used, if the content is not understood by or acceptable to the identity provider, then a
2136 `<Response>` message element MUST be returned with an error `<Status>`, and MAY contain a second-
2137 level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

2138 If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-`
2139 `format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any
2140 additional constraints due to the content of this element or the policies of the identity provider or principal.

2141 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates
2142 that the resulting assertion(s) MUST contain `<EncryptedID>` elements instead of plaintext. The
2143 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the
2144 requested subject.

2145 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an
2146 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly
2147 specific to the service provider) require that an encrypted identifier be used.

2148 Note that if the requester wishes to permit the identity provider to establish a new identifier for the principal
2149 if none exists, it MUST include a this element with the `AllowCreate` attribute set to "true". Otherwise,
2150 only a principal for whom the identity provider has previously established an identifier usable by the
2151 requester can be authenticated successfully. This is primarily useful in conjunction with the
2152 `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` `Format` value (see section 8.3.7).

2153 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
2154 complex type:

```
2155        <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>  
2156        <complexType name="NameIDPolicyType">  
2157            <attribute name="Format" type="anyURI" use="optional"/>  
2158            <attribute name="SPNameQualifier" type="string" use="optional"/>  
2159            <attribute name="AllowCreate" type="boolean" use="optional"/>  
2160        </complexType>
```

2161 3.4.1.2 Element `<Scoping>`

2162 The `<Scoping>` element specifies the identity providers trusted by the requester to authenticate the
2163 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
2164 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
2165 elements and attribute:

2166 ProxyCount [Optional]

2167 Specifies the number of proxying indirections permissible between the identity provider that receives
2168 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of
2169 zero permits no proxying, while omitting this attribute expresses no such restriction.

2170 <IDPList> [Optional]
 2171 An advisory list of identity providers and associated information that the requester deems acceptable
 2172 to respond to the request.

2173 <RequesterID> [Zero or More]
 2174 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate
 2175 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for a
 2176 description of entity identifiers.

2177 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
 2178 <Response> message with an error <Status> and a second-level <StatusCode> of
 2179 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or
 2180 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support
 2181 any of the specified identity providers.

2182 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```

2183 <element name="Scoping" type="samlp:ScopingType"/>
2184 <complexType name="ScopingType">
2185   <sequence>
2186     <element ref="samlp:IDPList" minOccurs="0"/>
2187     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2188   </sequence>
2189   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2190 </complexType>
2191 <element name="RequesterID" type="anyURI"/>
  
```

2192 3.4.1.3 Element <IDPList>

2193 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the
 2194 presenter. Its **IDPListType** complex type defines the following elements:

2195 <IDPEntry> [One or More]
 2196 Information about a single identity provider.

2197 <GetComplete> [Optional]
 2198 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to
 2199 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML
 2200 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>
 2201 element.

2202 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```

2203 <element name="IDPList" type="samlp:IDPListType"/>
2204 <complexType name="IDPListType">
2205   <sequence>
2206     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2207     <element ref="samlp:GetComplete" minOccurs="0"/>
2208   </sequence>
2209 </complexType>
2210 <element name="GetComplete" type="anyURI"/>
  
```

2211 3.4.1.3.1 Element <IDPEntry>

2212 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the
 2213 presenter. Its **IDPEntryType** complex type defines the following attributes:

2214 ProviderID [Required]
 2215 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2216 Name [Optional]

2217 A human-readable name for the identity provider.

2218 Loc [Optional]

2219 A URI reference representing the location of a profile-specific endpoint supporting the authentication
2220 request protocol. The binding to be used must be understood from the profile of use.

2221 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
2222 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2223 <complexType name="IDPEntryType">
2224   <attribute name="ProviderID" type="anyURI" use="required"/>
2225   <attribute name="Name" type="string" use="optional"/>
2226   <attribute name="Loc" type="anyURI" use="optional"/>
2227 </complexType>
```

2228 3.4.1.4 Processing Rules

2229 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
2230 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
2231 kinds of input and output are required or prohibited. The following processing rules apply as invariant
2232 behavior across any profile of this protocol exchange. All other processing rules associated with the
2233 underlying request and response messages **MUST** also be observed.

2234 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing
2235 one or more assertions that meet the specifications defined by the request, or with a <Response>
2236 message containing a <Status> describing the error that occurred. The responder **MAY** conduct
2237 additional message exchanges with the presenter as needed to initiate or complete the authentication
2238 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
2239 the next section, this includes proxying the request by directing the presenter to another identity provider
2240 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to
2241 authenticate the presenter to the original responder, in effect using SAML as the authentication
2242 mechanism.

2243 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if
2244 prevented from providing an assertion by policies in effect at the identity provider (for example the
2245 intended subject has prohibited the identity provider from providing assertions to the relying party), then it
2246 **MUST** return a <Response> with an error <Status>, and **MAY** return a second-level <StatusCode> of
2247 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
2248 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2249 If the <saml:Subject> element in the request is present, then the resulting assertions'
2250 <saml:Subject> **MUST strongly match** the request <saml:Subject>, as described in Section 3.3.4,
2251 except that the identifier **MAY** be in a different format if specified by <NameIDPolicy>. In such a case,
2252 the identifier's physical content **MAY** be different, but it **MUST** refer to the same principal.

2253 All of the content defined specifically within <AuthnRequest> is optional, although some may be required
2254 by certain profiles. In the absence of any specific content at all, the following behavior is implied:

- 2255 • The assertion(s) returned **MUST** contain a <saml:Subject> element that represents the
2256 presenter. The identifier type and format are determined by the identity provider. At least one
2257 statement in at least one assertion **MUST** be a <saml:AuthnStatement> that describes the
2258 authentication performed by the responder or authentication service associated with it.
- 2259 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the
2260 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation
2261 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.

- 2262 • The resulting assertion(s) MUST contain a `<saml:AudienceRestriction>` element
2263 referencing the requester as an acceptable relying party. Other audiences MAY be included as
2264 deemed appropriate by the identity provider.

2265 **3.4.1.5 Proxying**

2266 If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or
2267 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
2268 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
2269 `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in
2270 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
2271 identity provider.

2272 Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the
2273 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
2274 proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>`,
2275 completing the overall message exchange. Both the proxying and authenticating identity providers MAY
2276 include constraints on proxying activity in the messages and assertions they issue, as described in
2277 previous sections and below.

2278 The requester can influence proxy behavior by including a `<Scoping>` element where the provider sets a
2279 desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by
2280 including an ordered `<IDPList>` of preferred providers.

2281 An identity provider can control secondary use of its assertions by proxying identity providers using a
2282 `<ProxyRestriction>` element in the assertions it issues.

2283 **3.4.1.5.1 Proxying Processing Rules**

2284 An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is
2285 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2286 choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

2287 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity
2288 provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of
2289 `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly
2290 authenticate the presenter.

2291 If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying
2292 identity provider MUST include equivalent or stricter forms of all the information included in the original
2293 request (such as authentication context policy). Note, however, that the proxying provider is free to specify
2294 whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

2295 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
2296 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for
2297 example) will be honored by the authenticating provider.

2298 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less
2299 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new
2300 request SHOULD contain a `<ProxyCount>` attribute.

2301 If an `<IDPList>` was specified in the original request, the new request MUST also contain an
2302 `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the
2303 `<IDPList>`, but MUST NOT remove any from the list.

2304 The authentication request and response are processed in normal fashion, in accordance with the rules
2305 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
2306 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2307 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
2308 relevant information from the original assertion or non-SAML equivalent.
- 2309 • The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original
2310 requester 's preferences, as defined by its <NameIDPolicy> element.
- 2311 • The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext>
2312 element containing a <saml:AuthenticatingAuthority> element referencing the identity
2313 provider to which the proxying identity provider referred the presenter. If the original assertion
2314 contains <saml:AuthnContext> information that includes one or more
2315 <saml:AuthenticatingAuthority> elements, those elements SHOULD be included in the
2316 new assertion, with the new element placed after them.
- 2317 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
2318 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
2319 consistent over time across different requests. The value MUST not conflict with values used or
2320 generated by other SAML providers.
- 2321 • Any other <saml:AuthnContext> information MAY be copied, translated, or omitted in
2322 accordance with the policies of the proxying identity provider, provided that the original
2323 requirements dictated by the requester are met.

2324 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,
2325 and this request is equally or less strict than the original request (as determined by the proxying identity
2326 provider), the identity provider MAY skip the creation of a new <AuthnRequest> to the authenticating
2327 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML
2328 equivalent it received is still valid).

2329 **3.5 Artifact Resolution Protocol**

2330 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
2331 transported in a SAML binding by reference instead of by value. Both requests and responses can be
2332 obtained by reference using this specialized protocol. A message sender, instead of binding a message to
2333 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2334 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver
2335 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
2336 protocol to resolve the artifact into the original protocol message.

2337 The most common use for this mechanism is with bindings that cannot easily carry a message because of
2338 size constraints, or to enable a message to be communicated via a secure channel between the SAML
2339 requester and responder, avoiding the need for a signature.

2340 Depending on the characteristics of the underlying message being passed by reference, the artifact
2341 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2342 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
2343 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
2344 by any party.

2345 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
2346 as if the message so obtained had been sent originally in place of the artifact.

2347 3.5.1 Element <ArtifactResolve>

2348 The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an
2349 <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message.
2350 The original transmission of the artifact is governed by the specific protocol binding that is being used; see
2351 [SAMLBind] for more information on the use of artifacts in bindings.

2352 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
2353 protected by the protocol binding used to deliver the message.

2354 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
2355 adds the following element:

2356 <Artifact> [Required]

2357 The artifact value that the requester received and now wishes to translate into the protocol message it
2358 represents. See [SAMLBind] for specific artifact format information.

2359 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**
2360 complex type:

```
2361 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
2362 <complexType name="ArtifactResolveType">
2363   <complexContent>
2364     <extension base="samlp:RequestAbstractType">
2365       <sequence>
2366         <element ref="samlp:Artifact"/>
2367       </sequence>
2368     </extension>
2369   </complexContent>
2370 </complexType>
2371 <element name="Artifact" type="string"/>
```

2372 3.5.2 Element <ArtifactResponse>

2373 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
2374 message element. This element is of complex type **ArtifactResponseType**, which extends
2375 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2376 message being returned. This wrapped message element can be a request or a response.

2377 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2378 protected by the protocol binding used to deliver the message.

2379 The following schema fragment defines the <ArtifactResponse> element and its
2380 **ArtifactResponseType** complex type:

```
2381 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2382 <complexType name="ArtifactResponseType">
2383   <complexContent>
2384     <extension base="samlp:StatusResponseType">
2385       <sequence>
2386         <any namespace="##any" processContents="lax" minOccurs="0"/>
2387       </sequence>
2388     </extension>
2389   </complexContent>
2390 </complexType>
```

2391 3.5.3 Processing Rules

2392 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2393 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>

2394 element with no embedded message. In both cases, the <Status> element MUST include a
2395 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A
2396 response message with no embedded message inside it is termed an empty response in the remainder of
2397 this section.

2398 The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent
2399 request with the same artifact by any requester results in an empty response as described above.

2400 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY
2401 be intended for consumption by any party that receives it and can respond appropriately. In most other
2402 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2403 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2404 receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the
2405 original intended recipient, then the artifact issuer MUST return an empty response.

2406 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2407 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2408 return it in an <ArtifactResolve> message to the issuer.

2409 Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of
2410 the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message
2411 will contain its own message identifier, and in the case of an embedded response, may contain a different
2412 InResponseTo value that corresponds to the original request message to which the embedded message
2413 is responding.

2414 All other processing rules associated with the underlying request and response messages MUST be
2415 observed.

2416 **3.6 Name Identifier Management Protocol**

2417 After establishing a name identifier for a principal, an identity provider wishing to change the value and/or
2418 format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will
2419 no longer be used to refer to the principal, informs service providers of the change by sending them a
2420 <ManageNameIDRequest> message.

2421 A service provider also uses this message to register or change the SPProvidedID value to be included
2422 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
2423 identifier between itself and the identity provider.

2424 Note that this protocol is typically not used with "transient" name identifiers, since their value is not
2425 intended to be managed on a long term basis.

2426 **3.6.1 Element <ManageNameIDRequest>**

2427 A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name
2428 identifier or to indicate the termination of the use of a name identifier.

2429 The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity
2430 protected by the protocol binding used to deliver the message.

2431 This message has the complex type **ManageNameIDRequestType**, which extends
2432 **RequestAbstractType** and adds the following elements:

2433 <saml:NameID> OR <saml:EncryptedID> [Required]

2434 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
2435 principal as currently recognized by the identity and service providers prior to this request. (For more
2436 information on these elements, see Section 2.2.)

2437 <NewID> or <NewEncryptedID> or <Terminate> [Required]

2438 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2439 requesting provider concerning this principal, or an indication that the use of the old identifier has
2440 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
2441 appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the
2442 identity provider, the new value will appear in subsequent <NameID> elements as the element's
2443 content.

2444 The following schema fragment defines the <ManageNameIDRequest> element and its
2445 **ManageNameIDRequestType** complex type:

```
2446 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2447 <complexType name="ManageNameIDRequestType">
2448   <complexContent>
2449     <extension base="samlp:RequestAbstractType">
2450       <sequence>
2451         <choice>
2452           <element ref="saml:NameID"/>
2453           <element ref="saml:EncryptedID"/>
2454         </choice>
2455         <choice>
2456           <element ref="samlp:NewID"/>
2457           <element ref="samlp:NewEncryptedID"/>
2458           <element ref="samlp:Terminate"/>
2459         </choice>
2460       </sequence>
2461     </extension>
2462   </complexContent>
2463 </complexType>
2464 <element name="NewID" type="string"/>
2465 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
2466 <element name="Terminate" type="samlp:TerminateType"/>
2467 <complexType name="TerminateType"/>
```

2468 3.6.2 Element <ManageNameIDResponse>

2469 The recipient of a <ManageNameIDRequest> message MUST respond with a
2470 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional
2471 content.

2472 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity
2473 protected by the protocol binding used to deliver the message.

2474 The following schema fragment defines the <ManageNameIDResponse> element:

```
2475 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

2476 3.6.3 Processing Rules

2477 If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize,
2478 the responding provider MUST respond with an error <Status> and MAY respond with a second-level
2479 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2480 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the
2481 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of
2482 an identity provider) it will no longer issue assertions to the service provider about the principal. The
2483 receiving provider can perform any maintenance with the knowledge that the relationship represented by
2484 the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for
2485 whom a relationship has been terminated.

2486 If the service provider requests that its identifier for the principal be changed by including a <NewID> (or
2487 <NewEncryptedID>) element, the identity provider MUST include the element's content as the
2488 SPProvidedID when subsequently communicating to the service provider regarding this principal.

2489 If the identity provider requests that its identifier for the principal be changed by including a <NewID> (or
2490 <NewEncryptedID>) element, the service provider MUST use the element's content as the
2491 <saml:NameID> element content when subsequently communicating with the identity provider regarding
2492 this principal.

2493 Note that neither, either, or both of the original and new identifier MAY be encrypted (using the
2494 <EncryptedID> and <NewEncryptedID> elements).

2495 In any case, the <saml:NameID> content in the request and its associated SPProvidedID attribute
2496 MUST contain the most recent name identifier information established between the providers for the
2497 principal.

2498 In the case of an identifier with a Format of urn:oasis:names:tc:SAML:2.0:nameid-
2499 format:persistent, the NameQualifier attribute MUST contain the unique identifier of the identity
2500 provider that created the identifier. If the identifier was established between the identity provider and an
2501 affiliation group of which the service provider is a member, then the SPNameQualifier attribute MUST
2502 contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the
2503 service provider. These attributes MAY be omitted if they would otherwise match the value of the
2504 containing protocol message's <Issuer> element, but this is NOT RECOMMENDED due to the
2505 opportunity for confusion.

2506 Changes to these identifiers may take a potentially significant amount of time to propagate through the
2507 systems at both the requester and the responder. Implementations might wish to allow each party to
2508 accept either identifier for some period of time following the successful completion of a name identifier
2509 change. Not doing so could result in the inability of the principal to access resources.

2510 All other processing rules associated with the underlying request and response messages MUST be
2511 observed.

2512 **3.7 Single Logout Protocol**

2513 The single logout protocol provides a message exchange protocol by which all sessions provided by a
2514 particular session authority are near-simultaneously terminated. The single logout protocol is used either
2515 when a principal logs out at a session participant or when the principal logs out directly at the
2516 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
2517 the logout event can be indicated through the Reason attribute.

2518
2519 The principal may have established authenticated sessions with both the session authority and individual
2520 session participants, based on assertions containing authentication statements supplied by the session
2521 authority.

2522
2523 When the principal invokes the single logout process at a session participant, the session participant
2524 MUST send a <LogoutRequest> message to the session authority that provided the assertion
2525 containing the authentication statement related to that session at the session participant.

2526
2527 When either the principal invokes a logout at the session authority, or a session participant sends a logout
2528 request to the session authority specifying that principal, the session authority SHOULD send a
2529 <LogoutRequest> message to each session participant to which it provided assertions containing
2530 authentication statements under its current session with the principal, with the exception of the session
2531 participant that sent the <LogoutRequest> message to the session authority. It SHOULD attempt to
2532 contact as many of these participants as it can using this protocol, terminate its own session with the
2533 principal, and finally return a <LogoutResponse> message to the requesting session participant, if any.

2534 3.7.1 Element <LogoutRequest>

2535 A session participant or session authority sends a <LogoutRequest> message to indicate that a session
2536 has been terminated.

2537 The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected
2538 by the protocol binding used to deliver the message.

2539 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and
2540 adds the following elements and attributes:

2541 NotOnOrAfter [Optional]

2542 The time at which the request expires, after which the recipient may discard the message. The time
2543 value is encoded in UTC, as described in Section 1.3.3.

2544 Reason [Optional]

2545 An indication of the reason for the logout, in the form of a URI reference.

2546 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2547 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as
2548 currently recognized by the identity and service providers prior to this request. (For more information
2549 on this element, see Section 2.2.)

2550 <SessionIndex> [Optional]

2551 The identifier that indexes this session at the message recipient.

2552 The following schema fragment defines the <LogoutRequest> element and associated
2553 **LogoutRequestType** complex type:

```
2554 <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
2555 <complexType name="LogoutRequestType">
2556 <complexContent>
2557 <extension base="samlp:RequestAbstractType">
2558 <sequence>
2559 <choice>
2560 <element ref="saml:BaseID"/>
2561 <element ref="saml:NameID"/>
2562 <element ref="saml:EncryptedID"/>
2563 </choice>
2564 <element ref="samlp:SessionIndex" minOccurs="0"
2565 maxOccurs="unbounded"/>
2566 </sequence>
2567 <attribute name="Reason" type="string" use="optional"/>
2568 <attribute name="NotOnOrAfter" type="dateTime"
2569 use="optional"/>
2570 </extension>
2571 </complexContent>
2572 </complexType>
2573 <element name="SessionIndex" type="string"/>
```

2574 3.7.2 Element <LogoutResponse>

2575 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of
2576 type **StatusResponseType**, with no additional content specified.

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

2579 The following schema fragment defines the <LogoutResponse> element:

2580 `<element name="LogoutResponse" type="samlp:StatusResponseType"/>`

2581 3.7.3 Processing Rules

2582 The message sender MAY use the `Reason` attribute to indicate the reason for sending the
2583 `<LogoutRequest>`. The following values are defined by this specification for use by all message
2584 senders; other values MAY be agreed on between participants:

2585 `urn:oasis:names:tc:SAML:2.0:logout:user`

2586 Specifies that the message is being sent because the principal wishes to terminate the indicated
2587 session.

2588 `urn:oasis:names:tc:SAML:2.0:logout:admin`

2589 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2590 session for that principal.

2591 All other processing rules associated with the underlying request and response messages MUST be
2592 observed.

2593 Additional processing rules are provided in the following sections.

2594 3.7.3.1 Session Participant Rules

2595 When a session participant receives a `<LogoutRequest>` message, the session participant MUST
2596 authenticate the message. If the sender is the authority that provided an assertion containing an
2597 authentication statement linked to the principal's current session, the session participant MUST invalidate
2598 the principal's session(s) referred to by the `<saml:BaseID>`, `<saml:NameID>`, or
2599 `<saml:EncryptedID>` element, and any `<SessionIndex>` elements supplied in the message. If no
2600 `<SessionIndex>` elements are supplied, then all sessions associated with the principal MUST be
2601 invalidated.

2602
2603 The session participant MUST apply the logout request message to any assertion that meets the following
2604 conditions, even if the assertion arrives after the logout request:

- 2605 • The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or
2606 `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in section 3.3.4.
- 2607 • The `SessionIndex` attribute of one of the assertion's authentication statements matches one of
2608 the `<SessionIndex>` elements specified in the logout request, or the logout request contains no
2609 `<SessionIndex>` elements.
- 2610 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself
2611 (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject
2612 confirmation data).
- 2613 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on
2614 the message).

2615 **Note:** This rule is intended to prevent a situation in which a session participant receives a
2616 logout request targeted at a single, or multiple, assertion(s) (as identified by the
2617 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid –
2618 assertion(s) targeted by the logout request. It should honor the logout request until the
2619 logout request itself may be discarded (the `NotOnOrAfter` value on the request has
2620 been exceeded) or the assertion targeted by the logout request has been received and
2621 has been handled appropriately.

2622 3.7.3.2 Session Authority Rules

2623 When a session authority receives a `<LogoutRequest>` message, the session authority MUST
2624 authenticate the sender. If the sender is a session participant to which the session authority provided an
2625 assertion containing an authentication statement for the current session, then the session authority
2626 SHOULD do the following in the specified order:

- 2627 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session
2628 authority proxied the principal's authentication, unless the second authority is the originator of the
2629 `<LogoutRequest>`.
- 2630 • Send a `<LogoutRequest>` message to each session participant for which the session authority
2631 provided assertions in the current session, *other than* the originator of a current
2632 `<LogoutRequest>`.
- 2633 • Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`,
2634 or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout
2635 request message.

2636 If the session authority successfully terminates the principal's session with respect to itself, then it MUST
2637 respond to the original requester, if any, with a `<LogoutResponse>` message containing a top-level
2638 status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it MUST
2639 respond with a `<LogoutResponse>` message containing a top-level status code indicating the error.
2640 Thus, the top-level status indicates the state of the logout operation only with respect to the session
2641 authority itself.

2642 The session authority SHOULD attempt to contact each session participant using any applicable/usable
2643 protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because
2644 the original request takes place using a protocol binding that does not enable the logout to be propagated
2645 to all participants).

2646 In the event that not all session participants successfully respond to these `<LogoutRequest>` messages
2647 (or if not all participants can be contacted), then the session authority MUST include in its
2648 `<LogoutResponse>` message a second-level status code of
2649 `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session
2650 participants successfully responded with confirmation of the logout.

2651 Note that a session authority MAY initiate a logout for reasons other than having received a
2652 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 2653 • If some timeout period was agreed out-of-band with an individual session participant, the session
2654 authority MAY send a `<LogoutRequest>` to that individual participant alone.
- 2655 • An agreed global timeout period has been exceeded.
- 2656 • The principal or some other trusted entity has requested logout of the principal directly at the session
2657 authority.
- 2658 • The session authority has determined that the principal's credentials may have been compromised.

2659 When constructing a logout request message, the session authority MUST set the value of the
2660 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message,
2661 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time
2662 value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most
2663 recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout
2664 request).

2665 In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also
2666 available for use by the session authority only:

2667 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2668 Specifies that the message is being sent because of the global session timeout interval period
2669 being exceeded.

2670 urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2671 Specifies that the message is being sent because a timeout interval period agreed between a
2672 participant and the session authority has been exceeded.

2673 3.8 Name Identifier Mapping Protocol

2674 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2675 identifier for the same principal in a particular format or federation namespace, it can send a request to
2676 the identity provider using this protocol.

2677 For example, a service provider that wishes to communicate with another service provider with whom it
2678 does not share an identifier for the principal can use an identity provider that shares an identifier for the
2679 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,
2680 with which it can communicate with the second service provider.

2681 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
2682 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

2683 3.8.1 Element <NameIDMappingRequest>

2684 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2685 <NameIDMappingRequest> message. This message has the complex type
2686 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2687 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2688 The identifier and associated descriptive data that specify the principal as currently recognized by the
2689 requester and the responder. (For more information on this element, see Section 2.2.)

2690 <NameIDPolicy> [Required]

2691 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2692 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2693 binding used to deliver the message.

2694 The following schema fragment defines the <NameIDMappingRequest> element and its
2695 **NameIDMappingRequestType** complex type:

```
2696 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2697 <complexType name="NameIDMappingRequestType">
2698   <complexContent>
2699     <extension base="samlp:RequestAbstractType">
2700       <sequence>
2701         <choice>
2702           <element ref="saml:BaseID"/>
2703           <element ref="saml:NameID"/>
2704           <element ref="saml:EncryptedID"/>
2705         </choice>
2706         <element ref="samlp:NameIDPolicy"/>
2707       </sequence>
2708     </extension>
2709   </complexContent>
2710 </complexType>
```

2711 3.8.2 Element <NameIDMappingResponse>

2712 The recipient of a <NameIDMappingRequest> message MUST respond with a
2713 <NameIDMappingResponse> message. This message has the complex type
2714 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

2715 <saml:NameID> or <saml:EncryptedID> [Required]

2716 The identifier and associated attributes that specify the principal in the manner requested, usually in
2717 encrypted form. (For more information on this element, see Section 2.2.)

2718 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2719 binding used to deliver the message.

2720 The following schema fragment defines the <NameIDMappingResponse> element and its
2721 **NameIDMappingResponseType** complex type:

```
2722 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2723 <complexType name="NameIDMappingResponseType">
2724   <complexContent>
2725     <extension base="samlp:StatusResponseType">
2726       <choice>
2727         <element ref="saml:NameID"/>
2728         <element ref="saml:EncryptedID"/>
2729       </choice>
2730     </extension>
2731   </complexContent>
2732 </complexType>
```

2733 3.8.3 Processing Rules

2734 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2735 <Status> containing a second-level <StatusCode> of
2736 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2737 At the responder's discretion, the
2738 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to
2739 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2740 All other processing rules associated with the underlying request and response messages MUST be
2741 observed.

2742

4 SAML Versioning

2743
2744
2745
2746

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

2747
2748
2749

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

2750

$Major_B > Major_A$ OR (($Major_B = Major_A$) AND $Minor_B > Minor_A$)

2751

4.1 SAML Specification Set Version

2752
2753
2754
2755
2756
2757

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

2758
2759
2760
2761
2762

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

2763

4.1.1 Schema Version

2764
2765
2766
2767
2768

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

2769

4.1.2 SAML Assertion Version

2770
2771
2772
2773

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

2774
2775

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

2776

The following processing rules apply:

2777
2778

- A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion version number not supported by the authority.

2779
2780

- A SAML relying party MUST NOT process any assertion with a major assertion version number not supported by the relying party.

2781
2782

- A SAML relying party MAY process or MAY reject an assertion whose minor assertion version number is higher than the minor assertion version number supported by the relying party. However,

2783 all assertions that share a major assertion version number MUST share the same general
2784 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For
2785 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the
2786 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the
2787 likely effects of schema evolution.)

2788 **4.1.3 SAML Protocol Version**

2789 The various SAML protocols' request and response elements contain an attribute for expressing the major
2790 and minor version of the request or response message using a string of the form *Major.Minor*. Each
2791 version of the SAML specification set will be construed so as to document the syntax, semantics, and
2792 processing rules of the protocol messages of the same version. That is, specification set version 1.0
2793 describes request and response version V1.0, and so on.

2794 There is explicitly NO relationship between the protocol version and the target XML namespace specified
2795 for the schema definitions for that protocol version.

2796 The version numbers used in SAML protocol request and response elements will match for any particular
2797 revision of the SAML specification set.

2798 **4.1.3.1 Request Version**

2799 The following processing rules apply to requests:

- 2800 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2801 SAML requester and the SAML responder.
- 2802 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD
2803 assume that the responder supports requests with the highest request version supported by the
2804 requester.
- 2805 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version
2806 number matching a response version number that the requester does not support.
- 2807 • A SAML responder MUST reject any request with a major request version number not supported by
2808 the responder.
- 2809 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2810 higher than the highest supported request version that it supports. However, all requests that share
2811 a major request version number MUST share the same general processing rules and semantics,
2812 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the
2813 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill
2814 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

2815 **4.1.3.2 Response Version**

2816 The following processing rules apply to responses:

- 2817 • A SAML responder MUST NOT issue a response message with a response version number higher
2818 than the request version number of the corresponding request message.
- 2819 • A SAML responder MUST NOT issue a response message with a major response version number
2820 lower than the major request version number of the corresponding request message except to
2821 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2822 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
2823 top-level `<StatusCode>` value of
2824 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting

2825 one of the following second-level values:
2826 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,
2827 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, **OR**
2828 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

2829 **4.1.3.3 Permissible Version Combinations**

2830 Assertions of a particular major version appear only in response messages of the same major version, as
2831 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2832 example, a V1.1 assertion **MAY** appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2833 response message, if the appropriate assertion schema is referenced during namespace importation. But
2834 a V1.0 assertion **MUST NOT** appear in a V2.0 response message because they are of different major
2835 versions.

2836 **4.2 SAML Namespace Version**

2837 XML schema documents published as part of the specification set contain one or more target
2838 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct
2839 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that
2840 part of the specification.

2841 The namespace URI references defined by the specification set will generally contain version information
2842 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI **MUST** correspond
2843 to the major and minor version of the specification set in which the namespace is first introduced and
2844 defined. This information is not typically consumed by an XML processor, which treats the namespace
2845 opaquely, but is intended to communicate the relationship between the specification set and the
2846 namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that are
2847 listed in Section 8.

2848 As a general rule, implementers can expect the namespaces and the associated schema definitions
2849 defined by a major revision of the specification set to remain valid and stable across minor revisions of the
2850 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but
2851 this is expected to be rare. In such cases, the older namespaces and their associated definitions should
2852 be expected to remain valid until a major specification set revision.

2853 **4.2.1 Schema Evolution**

2854 In general, maintaining namespace stability while adding or changing the content of a schema are
2855 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2856 older implementations will react to any given change, making forward compatibility difficult to achieve.
2857 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
2858 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),
2859 implementations should expect forward-compatible schema changes in minor revisions, allowing new
2860 messages to validate against older schemas.

2861 Implementations **SHOULD** expect and be prepared to deal with new extensions and message types in
2862 accordance with the processing rules laid out for those types. Minor revisions **MAY** introduce new types
2863 that leverage the extension facilities described in Section . Older implementations **SHOULD** reject such
2864 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
2865 include new query, statement, or condition types.

2866

5 SAML and XML Signature Syntax and Processing

2867 SAML assertions and SAML protocol request and response messages may be signed, with the following
2868 benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the
2869 asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-
2870 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the
2871 message originator supports message integrity, authentication of message origin to a destination, and, if
2872 the signature is based on the originator's public-private key pair, non-repudiation of origin.

2873 A digital signature is not always required in SAML. For example, in some circumstances, signatures may
2874 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing
2875 protocol response message. "Inherited" signatures should be used with care when the contained object
2876 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context
2877 must be retained to allow validation, exposing the XML content and adding potentially unnecessary
2878 overhead. As another example, the SAML relying party or SAML requester may have obtained an
2879 assertion or protocol message from the SAML asserting party or SAML responder directly (with no
2880 intermediaries) through a secure channel, with the asserting party or SAML responder having
2881 authenticated to the relying party or SAML responder by some means other than a digital signature.

2882 Many different techniques are available for "direct" authentication and secure channel establishment
2883 between two parties. The list includes TLS/SSL (see [RFC 2246]/[SSL3]), HMAC, password-based
2884 mechanisms, and so on. In addition, the applicable security requirements depend on the communicating
2885 applications and the nature of the assertion or message transported. It is RECOMMENDED that, in all
2886 other contexts, digital signatures be used for assertions and request and response messages.
2887 Specifically:

- 2888 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting
2889 party SHOULD be signed by the SAML asserting party.
- 2890 • A SAML protocol message arriving at a destination from an entity other than the originating sender
2891 SHOULD be signed by the sender.

2892 Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that
2893 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are
2894 intended to be the primary SAML signature mechanism, but this specification attempts to ensure
2895 compatibility with profiles that may require other mechanisms.

2896 Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be
2897 enveloped.

2898 5.1 Signing Assertions

2899 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as
2900 described in Section 2.

2901 5.2 Request/Response Signing

2902 All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected
2903 in the schema as described in Section 3.

2904 5.3 Signature Inheritance

2905 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
2906 or a request or response, which may be signed. When a SAML assertion does not contain a
2907 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a

2908 <ds:Signature> element, and the signature applies to the <Assertion> element and all its children,
2909 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
2910 interpretation should be equivalent to the case where the assertion itself was signed with the same key
2911 and signature options.

2912 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
2913 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY
2914 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
2915 information from the surrounding context, but no such inheritance should be inferred unless specifically
2916 identified by the profile.

2917 **5.4 XML Signature Profile**

2918 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
2919 and many choices. This section details constraints on these facilities so that SAML processors do not
2920 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
2921 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the ID
2922 attribute on <Assertion> and the various request and response elements. These attributes are
2923 collectively referred to in this section as the identifier attributes.

2924 Note that this profile only applies to the use of the <ds:Signature> elements found directly within SAML
2925 assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to
2926 SAML content are free to define other approaches.

2927 **5.4.1 Signing Formats and Algorithms**

2928 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
2929 detached.

2930 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
2931 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
2932 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

2933 **5.4.2 References**

2934 SAML assertions and protocol messages MUST supply a value for the ID attribute on the root element of
2935 the assertion or protocol message being signed. The assertion's or protocol message's root element may
2936 or may not be the root element of the actual XML document containing the signed assertion or protocol
2937 message (e.g., it might be contained within a SOAP envelope).

2938 Signatures MUST contain a single <ds:Reference> containing a same-document reference to the ID
2939 attribute value of the root element of the assertion or protocol message being signed. For example, if the
2940 ID attribute value is "foo", then the URI attribute in the <ds:Reference> element MUST be "#foo".

2941 **5.4.3 Canonicalization Method**

2942 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
2943 both in the <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a
2944 <ds:Transform> algorithm. Use of Exclusive Canonicalization ensures that signatures created over
2945 SAML messages embedded in an XML context can be verified independent of that context.

2946 5.4.4 Transforms

2947 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
2948 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
2949 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
2950 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2951 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
2952 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can
2953 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
2954 applying the transforms manually to the content and reverifying the result as consisting of the same SAML
2955 message.

2956 5.4.5 KeyInfo

2957 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of
2958 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be
2959 absent.

2960 5.4.6 Example

2961 Following is an example of a signed response containing a signed assertion. Line breaks have been
2962 added for readability; the signatures are not valid and cannot be successfully verified.

```
2963 <Response  
2964   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"  
2965   ID="_c7055387-af61-4fce-8b98-e2927324b306"  
2966   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"  
2967   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">  
2968   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>  
2969   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
2970     <ds:SignedInfo>  
2971       <ds:CanonicalizationMethod  
2972         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2973       <ds:SignatureMethod  
2974         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
2975       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">  
2976         <ds:Transforms>  
2977           <ds:Transform  
2978             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-  
2979 signature" />  
2980           <ds:Transform  
2981             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">  
2982             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"  
2983               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2984           </ds:Transform>  
2985         </ds:Transforms>  
2986         <ds:DigestMethod  
2987           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
2988         <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>  
2989       </ds:Reference>  
2990     </ds:SignedInfo>  
2991     <ds:SignatureValue>  
2992       x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5  
2993       EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWaptaKlywS7gFgsD01qjyen3CP+m3D  
2994       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=  
2995     </ds:SignatureValue>  
2996     <ds:KeyInfo>  
2997       <ds:X509Data>  
2998         <ds:X509Certificate>  
2999         MIIcYjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgkxCzAJBgNVBAYTAlVT
```

```

3000 MRIwEAYDVQQIEwLXaXNjb25zaW4xEDAObgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
3001 F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLLEyJEaXZpc2lvbiBvZiBJ
3002 bmZvcmlhdGlviBUZWNobm9sb2d5MSUwIwYDVQQDEExIRVBLSSBTZXJ2ZXIgc0Eg
3003 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowYsxCz
3004 AJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMjQW5uIEFy
3005 Ym9yMQ4wDAYDVQKKEwVQ0FJRDECMBoGA1UEAxMTc2hpYjEuaW50ZXJuzXQyLmVkdTEn
3006 MCUGCSqGSIB3DQEBQUAA4GNADCBiQKbGQDZSAb2sXvHAXnXVIVT8vuRay+x50z7GJj
3007 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAoAPSZB113R6+KYiE7x4XAWIrcP+
3008 c2MZVeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
3009 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3010 hkiG9w0BAQQFAAOBqQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3011 qgi7lFV6MDkHmTvtqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3012 8I3bsbmRAUg4UP9hH6ABVq4KQKMKnxu1xQxLhpR1yLGPdiowMNTREg8cCx3w/w==
3013 </ds:X509Certificate>
3014 </ds:X509Data>
3015 </ds:KeyInfo>
3016 </ds:Signature>
3017 </ds:Signature>
3018 <Status>
3019 <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
3020 </Status>
3021 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3022 IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3023 xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3024 <Issuer>https://www.opensaml.org/IDP</Issuer>
3025 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3026 <ds:SignedInfo>
3027 <ds:CanonicalizationMethod
3028 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3029 <ds:SignatureMethod
3030 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
3031 <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3032 <ds:Transforms>
3033 <ds:Transform
3034 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3035 signature"/>
3036 <ds:Transform
3037 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3038 <InclusiveNamespaces
3039 PrefixList="#default saml ds xs xsi"
3040 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#">
3041 </ds:Transform>
3042 </ds:Transforms>
3043 <ds:DigestMethod
3044 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
3045 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3046 </ds:Reference>
3047 </ds:SignedInfo>
3048 <ds:SignatureValue>
3049 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3050 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
3051 MwuL/cBUj20tBZOQMFN7jQ9YB7klIz3RqVL+wNmeWI4=
3052 </ds:SignatureValue>
3053 <ds:KeyInfo>
3054 <ds:X509Data>
3055 <ds:X509Certificate>
3056 MIICyCAjOgAwIBAgICAnUdQYJKoZIhvcNAQEEBQAwwgAKCzAJBgNVBAYTA1VT
3057 MRIwEAYDVQQIEwLXaXNjb25zaW4xEDAObgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
3058 F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLLEyJEaXZpc2lvbiBvZiBJ
3059 bmZvcmlhdGlviBUZWNobm9sb2d5MSUwIwYDVQQDEExIRVBLSSBTZXJ2ZXIgc0Eg
3060 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowYsxCz
3061 AJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMjQW5uIEFy
3062 Ym9yMQ4wDAYDVQKKEwVQ0FJRDECMBoGA1UEAxMTc2hpYjEuaW50ZXJuzXQyLmVkdTEn
3063 MCUGCSqGSIB3DQEBQUAA4GNADCBiQKbGQDZSAb2sXvHAXnXVIVT8vuRay+x50z7GJj
3064 dTEnMCUGCSqGSIB3DQEBQUAA4GNADCBiQKbGQDZSAb2sXvHAXnXVIVT8vuRay+x50z7GJj
3065 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAoAPSZB113R6+KYiE7x4XAWIrcP+

```

```

3066      c2MZVeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
3067      pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAWIFoDANBgkq
3068      hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2N
3069      qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
3070      8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTreG8cCx3w/w==
3071      </ds:X509Certificate>
3072      </ds:X509Data>
3073      </ds:KeyInfo>
3074    </ds:Signature>
3075    <Subject>
3076      <NameID
3077        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3078        scott@example.org
3079      </NameID>
3080      <SubjectConfirmation
3081        Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3082    </Subject>
3083    <Conditions NotBefore="2003-04-17T00:46:02Z"
3084      NotOnOrAfter="2003-04-17T00:51:02Z">
3085      <AudienceRestriction>
3086        <Audience>http://www.opensaml.org/SP</Audience>
3087      </AudienceRestriction>
3088    </Conditions>
3089    <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3090      <AuthnContext>
3091        <AuthnContextClassRef>
3092          urn:oasis:names:tc:SAML:2.0:ac:classes:Password
3093        </AuthnContextClassRef>
3094      </AuthnContext>
3095    </AuthnStatement>
3096  </Assertion>
3097 </Response>

```

3098

6 SAML and XML Encryption Syntax and Processing

3099 Encryption is used as the means to implement confidentiality. The most common motives for
3100 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for
3101 competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness
3102 of some other security mechanism. For example, a secret password or key may be encrypted.

3103 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 3104 • Communications confidentiality may be provided by mechanisms associated with a particular
3105 binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS (see
3106 [RFC 2246]/[SSL3]) or SOAP Message Security mechanisms for confidentiality.
- 3107 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`
3108 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be
3109 encrypted.
- 3110 • An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- 3111 • The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- 3112 • An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

3113 6.1 General Considerations

3114 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use
3115 of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace
3116 the cleartext information in the same location within the XML instance. The `<EncryptedData>` element's
3117 `Type` attribute SHOULD be used and, if it is present, MUST have the value
3118 `http://www.w3.org/2001/04/xmlenc#Element`.

3119 Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The
3120 SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

3121 6.2 Combining Signatures and Encryption

3122 Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and
3123 encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in
3124 the reverse order that signing and encryption were performed.

- 3125 • When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and
3126 placed within the `<Assertion>` element before the element is encrypted.
- 3127 • When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be
3128 performed first and then the signature calculated over the assertion or message containing the
3129 encrypted element.

3130 7 SAML Extensibility

3131 SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas.
3132 An example of an application that extends SAML assertions is the Liberty Protocols and Schema
3133 Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions
3134 and protocols.

3135 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can
3136 be combined with extensions to put the SAML framework to new uses.

3137 7.1 Schema Extension

3138 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
3139 elements can serve as the head element of a substitution group. However, SAML types are not defined as
3140 *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that
3141 extensions are typically defined only as types rather than elements, and are included in SAML instances
3142 by means of an `xsi:type` attribute.

3143 The following sections discuss only elements and types that have been specifically designed to support
3144 extensibility.

3145 7.1.1 Assertion Schema Extension

3146 The SAML assertion schema (see [SAML-XSD]) is designed to permit separate processing of the
3147 assertion package and the statements it contains, if the extension mechanism is used for either part.

3148 The following elements are intended specifically for use as extension points in an extension schema; their
3149 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 3150 • `<BaseID>` and **BaseIDAbstractType**
- 3151 • `<Condition>` and **ConditionAbstractType**
- 3152 • `<Statement>` and **StatementAbstractType**

3153 The following constructs that are directly usable as part of SAML are particularly interesting targets for
3154 extension:

- 3155 • `<AuthnStatement>` and **AuthnStatementType**
- 3156 • `<AttributeStatement>` and **AttributeStatementType**
- 3157 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
- 3158 • `<AudienceRestriction>` and **AudienceRestrictionType**
- 3159 • `<ProxyRestriction>` and **ProxyRestrictionType**
- 3160 • `<OneTimeUse>` and **OneTimeUseType**

3161 7.1.2 Protocol Schema Extension

3162 The following SAML protocol elements are intended specifically for use as extension points in an
3163 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
3164 type:

- 3165 • `<Request>` and **RequestAbstractType**

3166 • <SubjectQuery> and **SubjectQueryAbstractType**

3167 The following constructs that are directly usable as part of SAML are particularly interesting targets for
3168 extension:

3169 • <AuthnQuery> and **AuthnQueryType**

3170 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**

3171 • <AttributeQuery> and **AttributeQueryType**

3172 • **StatusResponseType**

3173 7.2 Schema Wildcard Extension Points

3174 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes
3175 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension
3176 schema.

3177 7.2.1 Assertion Extension Points

3178 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

3179 • <SubjectConfirmationData>: Uses **xs:anyType**, which allows any sub-elements and
3180 attributes.

3181 • <AuthnContextDecl>: Uses **xs:anyType**, which allows any sub-elements and attributes.

3182 • <AttributeValue>: Uses **xs:anyType**, which allows any sub-elements and attributes.

3183 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other
3184 namespaces with lax schema validation processing.

3185 The following constructs in the assertion schema allow arbitrary global attributes:

3186 • <Attribute> and **AttributeType**

3187 7.2.2 Protocol Extension Points

3188 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

3189 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema
3190 validation processing.

3191 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax
3192 schema validation processing.

3193 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with
3194 lax schema validation processing. (It is specifically intended to carry a SAML request or response
3195 message element, however.)

3196 **7.3 Identifier Extension**

3197 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
3198 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
3199 However, it is always possible to define additional URI-based identifiers for these purposes. It is
3200 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should
3201 the meaning of a given URI used as such an identifier significantly change, or be used to mean two
3202 different things.

3203

8 SAML-Defined Identifiers

3204 The following sections define URI-based identifiers for common resource access actions, subject name
3205 identifier formats, and attribute name formats.

3206 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of
3207 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
3208 have one of the following stems, according to the specification set version in which they were first
3209 introduced:

```
3210 urn:oasis:names:tc:SAML:1.0:  
3211 urn:oasis:names:tc:SAML:1.1:  
3212 urn:oasis:names:tc:SAML:2.0:
```

8.1 Action Namespace Identifiers

3214 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to
3215 common sets of actions to perform on resources.

8.1.1 Read/Write/Execute/Delete/Control

3217 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

3218 Defined actions:

3219 `Read Write Execute Delete Control`

3220 These actions are interpreted as follows:

3221 `Read`

3222 The subject may read the resource.

3223 `Write`

3224 The subject may modify the resource.

3225 `Execute`

3226 The subject may execute the resource.

3227 `Delete`

3228 The subject may delete the resource.

3229 `Control`

3230 The subject may specify the access control policy for the resource.

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

3232 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

3233 Defined actions:

3234 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

3235 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions
3236 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
3237 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is
3238 affirmatively denied read permission.

3239 A SAML authority MUST NOT authorize both an action and its negated form.

3240 **8.1.3 Get/Head/Put/Post**

3241 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3242 Defined actions:

3243 GET HEAD PUT POST

3244 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
3245 the GET action on a resource is authorized to retrieve it.

3246 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST
3247 actions to the write permission. The correspondence is not exact however since an HTTP GET operation
3248 may cause data to be modified and a POST operation may cause modification to a resource other than
3249 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

3250 **8.1.4 UNIX File Permissions**

3251 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3252 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3253 The action string is a four-digit numeric code:

3254 *extended user group world*

3255 Where the *extended* access permission has the value

3256 +2 if sgid is set

3257 +4 if suid is set

3258 The *user group* and *world* access permissions have the value

3259 +1 if execute permission is granted

3260 +2 if write permission is granted

3261 +4 if read permission is granted

3262 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
3263 and execute; and world read.

3264 **8.2 Attribute Name Format Identifiers**

3265 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex
3266 type to refer to the classification of the attribute name for purposes of interpreting the name.

3267 **8.2.1 Unspecified**

3268 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3269 The interpretation of the attribute name is left to individual implementations.

3270 8.2.2 URI Reference

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

3272 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML
3273 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-
3274 specific. See [SAMLProf] for attribute profiles that make use of this identifier.

3275 8.2.3 Basic

3276 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

3277 The class of strings acceptable as the attribute name **MUST** be drawn from the set of values belonging to
3278 the primitive type **xs:Name** as defined in [Schema2] §3.3.6. See [SAMLProf] for attribute profiles that
3279 make use of this identifier.

3280 8.3 Name Identifier Format Identifiers

3281 The following identifiers **MAY** be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
3282 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the
3283 associated processing rules, if any.

3284 **Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for
3285 SAML V2.0.

3286 8.3.1 Unspecified

3287 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3288 The interpretation of the content of the element is left to individual implementations.

3289 8.3.2 Email Address

3290 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3291 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
3292 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that
3293 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in
3294 parentheses) after it, and is not surrounded by "<" and ">".

3295 8.3.3 X.509 Subject Name

3296 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3297 Indicates that the content of the element is in the form specified for the contents of the
3298 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
3299 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
3300 differ from the rules given in IETF RFC 2253 [RFC 2253].

3301 8.3.4 Windows Domain Qualified Name

3302 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3303 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3304 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator
3305 MAY be omitted.

3306 **8.3.5 Kerberos Principal Name**

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

3308 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3309 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
3310 realm are described in IETF RFC 1510 [RFC 1510].

3311 **8.3.6 Entity Identifier**

3312 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3313 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3314 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service
3315 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to
3316 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make
3317 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be
3318 used in other elements and attributes whose purpose is to identify a system entity in various protocol
3319 exchanges.

3320 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3321 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3322 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

3323 **8.3.7 Persistent Identifier**

3324 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3325 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3326 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
3327 generated by identity providers MUST be constructed using pseudo-random values that have no
3328 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
3329 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
3330 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3331 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity
3332 provider that generated the identifier (see section 8.3.6). It MAY be omitted if the value can be derived
3333 from the context of the message containing the element, such as the issuer of a protocol message or an
3334 assertion containing the identifier in its subject. Note that a different system entity might later issue its own
3335 protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in
3336 this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be
3337 omitted in such a case).

3338 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service
3339 provider or affiliation of providers for whom the identifier was generated (see section 8.3.6). It MAY be
3340 omitted if the element is contained in a message intended only for consumption directly by the service
3341 provider, and the value would be the unique identifier of that service provider.

3342 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most
3343 recently set by the service provider or affiliation, if any (see section 3.6). If no such identifier has been
3344 established, then the attribute MUST be omitted.

3345 Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be shared
3346 in clear text with providers other than the providers that have established the shared identifier.
3347 Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and
3348 protections. Deployments without such requirements are free to use other kinds of identifiers in their
3349 SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

3350 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3351 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3352 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3353 different and does not affect the behavior of the identity provider or any processing rules specific to
3354 persistent identifiers in the protocols defined in this specification.

3355 Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of
3356 creation, but not of use. If a persistent identifier is created by a particular identity provider, the
3357 `NameQualifier` attribute value is permanently established at that time. If a service provider that receives
3358 such an identifier takes on the role of an identity provider and issues its own assertion containing that
3359 identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It
3360 might alternatively choose to create its own persistent identifier to represent the principal and link the two
3361 values. This is a deployment decision.

3362 **8.3.8 Transient Identifier**

3363 **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

3364 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3365 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3366 accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of
3367 256 characters.

3368 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3369 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
3370 accordance with the rules specified in Section 8.3.7.

3371 **8.4 Consent Identifiers**

3372 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** and
3373 **StatusResponseType** complex types to communicate whether a principal gave consent, and under what
3374 conditions, for the message.

3375 **8.4.1 Unspecified**

3376 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

3377 No claim as to principal consent is being made.

3378 **8.4.2 Obtained**

3379 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

3380 Indicates that a principal's consent has been obtained by the issuer of the message.

3381 **8.4.3 Prior**

3382 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`

3383 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to
3384 the action that initiated the message.

3385 **8.4.4 Implicit**

3386 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3387 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the
3388 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically
3389 more proximal to the action in time and presentation than prior consent, such as part of a session of
3390 activities.

3391 **8.4.5 Explicit**

3392 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3393 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the
3394 action that initiated the message.

3395 **8.4.6 Unavailable**

3396 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3397 Indicates that the issuer of the message did not obtain consent.

3398 **8.4.7 Inapplicable**

3399 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3400 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

9 References

3401

3402 The following works are cited in the body of this specification.

9.1 Normative References

- 3404 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web
3405 Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3406 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
3407 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.
3408 Note that this specification normatively references [Schema2], listed below.
- 3409 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium
3410 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3411 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World
3412 Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3413 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing,
3414 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web
3415 Consortium. Note that this specification normatively references [XMLEnc-XSD],
3416 listed below.
- 3417 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium.
3418 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3419 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January
3420 1999. <http://www.w3.org/TR/REC-xml-names>.
- 3421 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web
3422 Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this
3423 specification normatively references [XMLSig-XSD], listed below.
- 3424 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium.
3425 [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)
3426 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

9.2 Non-Normative References

- 3427
- 3428 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty
3429 Alliance Project, January 2003,
3430 [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)
3431 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3432 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF
3433 RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3434 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
3435 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- 3436 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
3437 <http://www.ietf.org/rfc/rfc2246.txt>.
- 3438 **[RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*
3439 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.
3440 <http://www.ietf.org/rfc/rfc2253.txt>.
- 3441 **[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF
3442 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.
- 3443 **[RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.
3444 <http://www.ietf.org/rfc/rfc2822.txt>.

3445	[RFC 3075]	D. Eastlake, J. Reagle, D. Solo. <i>XML-Signature Syntax and Processing</i> . IETF RFC 3075, March 2001. http://www.ietf.org/rfc/rfc3075.txt .
3446		
3447	[RFC 3513]	R. Hinden, S. Deering, <i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i> . IETF RFC 3513, April 2003. http://www.ietf.org/rfc/rfc3513.txt .
3448		
3449	[SAMLAuthnCxt]	J. Kemp et al., <i>Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-authn-context-2.0-cd-04. See http://www.oasis-open.org/committees/security/ .
3450		
3451		
3452	[SAMLBind]	S. Cantor et al., <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-bindings-2.0-cd-04. See http://www.oasis-open.org/committees/security/ .
3453		
3454		
3455	[SAMLConform]	P. Mishra et al., <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-conformance-2.0-cd-04. http://www.oasis-open.org/committees/security/ .
3456		
3457		
3458	[SAMLGloss]	J. Hodges et al., <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-glossary-2.0-cd-04. See http://www.oasis-open.org/committees/security/ .
3459		
3460		
3461	[SAMLMeta]	S. Cantor et al., <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-metadata-2.0-cd-04. See http://www.oasis-open.org/committees/security/ .
3462		
3463		
3464	[SAMLXSD]	S. Cantor et al., SAML protocols schema. OASIS SSTC, January 2005. Document ID sstc-saml-schema-protocol-2.0. See http://www.oasis-open.org/committees/security/ .
3465		
3466		
3467	[SAMLProf]	S. Cantor et al., <i>Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-profiles-2.0-cd-04. See http://www.oasis-open.org/committees/security/ .
3468		
3469		
3470	[SAMLSecure]	F. Hirsch et al., <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, January 2005. Document ID sstc-saml-sec-consider-2.0-cd-04. See http://www.oasis-open.org/committees/security/ .
3471		
3472		
3473		
3474	[SAMLTechOvw]	J. Hughes et al. SAML Technical Overview. OASIS, July 2004. Document ID oasis-sstc-saml-tech-overview-2.0. http://www.oasis-open.org/committees/security/ .
3475		
3476		
3477	[SAMLXSD]	S. Cantor et al., SAML assertions schema. OASIS SSTC, January 2005. Document ID sstc-saml-schema-assertion-2.0. See http://www.oasis-open.org/committees/security/ .
3478		
3479		
3480	[SSL3]	A. Frier et al., <i>The SSL 3.0 Protocol</i> , Netscape Communications Corp, November 1996.
3481		
3482	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
3483		
3484	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. http://www.w3.org/TR/WD-charreq .
3485		
3486	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0: Normalization</i> . World Wide Web Consortium, February 2004. http://www.w3.org/TR/charmod-norm/ .
3487		
3488	[XACML]	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See http://www.oasis-open.org/committees/xacml .
3489		
3490	[XML-ID]	J. Marsh et al., <i>xml:id Version 1.0</i> , W3C, April 2004. http://www.w3.org/TR/xml-id/ .
3491		

Appendix A. Acknowledgments

3493 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
3494 Committee, whose voting members at the time of publication were:

- 3495 • Conor Cahill, AOL
- 3496 • John Hughes, Atos Origin
- 3497 • Hal Lockhart, BEA Systems
- 3498 • Mike Beach, Boeing
- 3499 • Rebekah Metz, Booz Allen Hamilton
- 3500 • Rick Randall, Booz Allen Hamilton
- 3501 • Ronald Jacobson, Computer Associates
- 3502 • Carolina Canales-Valenzuela, Ericsson
- 3503 • Dana Kaufman, Forum Systems
- 3504 • Irving Reid, Hewlett-Packard
- 3505 • Paula Austel, IBM
- 3506 • Michael McIntosh, IBM
- 3507 • Anthony Nadalin, IBM
- 3508 • Nick Ragouzis, Individual
- 3509 • Scott Cantor, Internet2
- 3510 • Bob Morgan, Internet2
- 3511 • Peter Davis, Neustar
- 3512 • Jeff Hodges, Neustar
- 3513 • Frederick Hirsch, Nokia
- 3514 • Senthil Sengodan, Nokia
- 3515 • Abbie Barbir, Nortel Networks
- 3516 • Scott Kiestler, Novell
- 3517 • Cameron Morris, Novell
- 3518 • Paul Madsen, NTT
- 3519 • Steve Anderson, OpenNetwork
- 3520 • Ari Kermaier, Oracle
- 3521 • Vamsi Motukuru, Oracle
- 3522 • Darren Platt, Ping Identity
- 3523 • Prateek Mishra, Principal Identity
- 3524 • Jim Lien, RSA Security
- 3525 • John Linn, RSA Security
- 3526 • Rob Philpott, RSA Security
- 3527 • Dipak Chopra, SAP
- 3528 • Jahan Moreh, Sigaba
- 3529 • Bhavna Bhatnagar, Sun Microsystems
- 3530 • Eve Maler, Sun Microsystems
- 3531 • Ronald Monzillo, Sun Microsystems
- 3532 • Emily Xu, Sun Microsystems
- 3533 • Greg Whitehead, Trustgenix

3534 The editors also would like to acknowledge the following people for their contributions to previous versions
3535 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 • Carlisle Adams, Entrust
- 3541 • Tim Moses, Entrust
- 3542 • Nigel Edwards, Hewlett-Packard
- 3543 • Joe Pato, Hewlett-Packard
- 3544 • Bob Blakley, IBM
- 3545 • Marlena Erdos, IBM
- 3546 • Marc Chanliau, Netegrity
- 3547 • Chris McLaren, Netegrity
- 3548 • Lynne Rosenthal, NIST
- 3549 • Mark Skall, NIST
- 3550 • Simon Godik, Overxeer
- 3551 • Charles Norwood, SAIC
- 3552 • Evan Prodromou, Securant
- 3553 • Robert Griffin, RSA Security (former editor)
- 3554 • Sai Allarvarpu, Sun Microsystems
- 3555 • Chris Ferris, Sun Microsystems
- 3556 • Mike Myers, Traceroute Security
- 3557 • Phillip Hallam-Baker, VeriSign (former editor)
- 3558 • James Vanderbeek, Vodafone
- 3559 • Mark O'Neill, Vordel
- 3560 • Tony Palmer, Vordel

3561

3562 Finally, the editors wish to acknowledge the following people for their contributions of material used as
3563 input to the OASIS Security Assertions Markup Language specifications:

- 3564 • Thomas Gross, IBM
- 3565 • Birgit Pfitzmann, IBM

3566

Appendix B. Notices

3567 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3568 might be claimed to pertain to the implementation or use of the technology described in this document or
3569 the extent to which any license under such rights might or might not be available; neither does it represent
3570 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
3571 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
3572 available for publication and any assurances of licenses to be made available, or the result of an attempt
3573 made to obtain a general license or permission for the use of such proprietary rights by implementors or
3574 users of this specification, can be obtained from the OASIS Executive Director.

3575 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
3576 other proprietary rights which may cover technology that may be required to implement this specification.
3577 Please address the information to the OASIS Executive Director.

3578 **Copyright © OASIS Open 2005. All Rights Reserved.**

3579 This document and translations of it may be copied and furnished to others, and derivative works that
3580 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
3581 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
3582 this paragraph are included on all such copies and derivative works. However, this document itself may
3583 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
3584 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
3585 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
3586 into languages other than English.

3587 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3588 or assigns.

3589 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3590 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3591 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3592 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.