



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

~~OASIS Standard Working Draft, 15 March 2005~~
August 2006

Document identifier:

~~saml-core-2.0-essstc-saml-core-errata-2.0-wd-00~~

Location:

~~http://docs.oasis-open.org/security/saml/v2.0/http://www.oasis-open.org/committees/documents.php?wg_abbrev=security~~

Editors:

Scott Cantor, Internet2
John Kemp, Nokia
Rob Philpott, RSA Security
[Jahan Moreh, Sigaba \(errata document editor\)](#)
Eve Maler, Sun Microsystems [\(errata composite document editor\)](#)

SAML V2.0 Contributors:

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

41 John Linn, RSA Security
42 Rob Philpott, RSA Security
43 Jahan Moreh, Sigaba
44 Anne Anderson, Sun Microsystems
45 Eve Maler, Sun Microsystems
46 Ron Monzillo, Sun Microsystems
47 Greg Whitehead, Trustgenix

48 **Abstract:**

49 The SAML V2.0 Assertions and Protocols This specification defines the syntax and semantics for
50 XML-encoded assertions about authentication, attributes, and authorization, and for the protocols
51 that convey this information. This document, known as an “errata composite”, combines
52 corrections to reported errata with the original specification text. By design, the corrections are
53 limited to clarifications of ambiguous or conflicting specification text. This document shows
54 deletions from the original specification as struck-through text, and additions as blue underlined
55 text. The “[Enn]” and “[PEnn]” designations embedded in the text refer to particular errata and
56 their dispositions.

57 **Status:**

58 Although this may look similar to the original This is an **OASIS Standard** document produced by
59 the Security Services Technical Committee. ~~It was and~~ approved by the OASIS membership on
60 1 March 2005, this errata composite document is a **non-normative working draft**. **N.B.:** The
61 SAML V2.0 errata document and the entire set of errata composite documents, including this
62 one, are *not* on an OASIS Standard track, but the text changes proposed here may ultimately
63 find their way into a future standards-track SAML specification.

64 This document includes errata corrections through revision 32 of the errata document, including
65 E1, PE6, PE8, PE10, PE12, PE13, PE14, PE15, PE30, PE36, PE38, PE43, PE45, PE46, PE47.

66 Committee members should submit comments and potential errata to the security-
67 services@lists.oasis-open.org list. Others should submit them by ~~filling out the web form~~
68 ~~located following the instructions~~ at http://www.oasis-
69 open.org/committees/comments/form.php?wg_abbrev=security. ~~The committee will publish on~~
70 ~~its web page (http://www.oasis-open.org/committees/security) a catalog of any changes made to~~
71 ~~this document as a result of comments.~~

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

Table of Contents

77	1 Introduction.....	7
78	1.1 Notation.....	7
79	1.2 Schema Organization and Namespaces.....	8
80	1.3 Common Data Types.....	8
81	1.3.1 String Values.....	8
82	1.3.2 URI Values.....	9
83	1.3.3 Time Values.....	9
84	1.3.4 ID and ID Reference Values.....	9
85	2 SAML Assertions.....	11
86	2.1 Schema Header and Namespace Declarations.....	11
87	2.2 Name Identifiers.....	12
88	2.2.1 Element <BaseID>.....	12
89	2.2.2 Complex Type NameIDType.....	13
90	2.2.3 Element <NameID>.....	14
91	2.2.4 Element <EncryptedID>.....	14
92	2.2.5 Element <Issuer>.....	15
93	2.3 Assertions.....	15
94	2.3.1 Element <AssertionIDRef>.....	15
95	2.3.2 Element <AssertionURIRef>.....	15
96	2.3.3 Element <Assertion>.....	15
97	2.3.4 Element <EncryptedAssertion>.....	17
98	2.4 Subjects.....	18
99	2.4.1 Element <Subject>.....	18
100	2.4.1.1 Element <SubjectConfirmation>.....	18
101	2.4.1.2 Element <SubjectConfirmationData>.....	19
102	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	21
103	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
104	2.5 Conditions.....	21
105	2.5.1 Element <Conditions>.....	21
106	2.5.1.1 General Processing Rules.....	22
107	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
108	2.5.1.3 Element <Condition>.....	23
109	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	23
110	2.5.1.5 Element <OneTimeUse>.....	24
111	2.5.1.6 Element <ProxyRestriction>.....	25
112	2.6 Advice.....	26
113	2.6.1 Element <Advice>.....	26
114	2.7 Statements.....	26
115	2.7.1 Element <Statement>.....	27
116	2.7.2 Element <AuthnStatement>.....	27
117	2.7.2.1 Element <SubjectLocality>.....	28
118	2.7.2.2 Element <AuthnContext>.....	28
119	2.7.3 Element <AttributeStatement>.....	29
120	2.7.3.1 Element <Attribute>.....	30
121	2.7.3.1.1 Element <AttributeValue>.....	31
122	2.7.3.2 Element <EncryptedAttribute>.....	31
123	2.7.4 Element <AuthzDecisionStatement>.....	32
124	2.7.4.1 Simple Type DecisionType.....	33

125	2.7.4.2 Element <Action>.....	34
126	2.7.4.3 Element <Evidence>.....	34
127	3 SAML Protocols.....	36
128	3.1 Schema Header and Namespace Declarations.....	36
129	3.2 Requests and Responses.....	37
130	3.2.1 Complex Type RequestAbstractType.....	37
131	3.2.2 Complex Type StatusResponseType.....	39
132	3.2.2.1 Element <Status>.....	40
133	3.2.2.2 Element <StatusCode>.....	41
134	3.2.2.3 Element <StatusMessage>.....	43
135	3.2.2.4 Element <StatusDetail>.....	43
136	3.3 Assertion Query and Request Protocol.....	43
137	3.3.1 Element <AssertionIDRequest>.....	43
138	3.3.2 Queries.....	44
139	3.3.2.1 Element <SubjectQuery>.....	44
140	3.3.2.2 Element <AuthnQuery>.....	44
141	3.3.2.2.1 Element <RequestedAuthnContext>.....	45
142	3.3.2.3 Element <AttributeQuery>.....	46
143	3.3.2.4 Element <AuthzDecisionQuery>.....	47
144	3.3.3 Element <Response>.....	48
145	3.3.4 Processing Rules.....	48
146	3.4 Authentication Request Protocol.....	49
147	3.4.1 Element <AuthnRequest>.....	49
148	3.4.1.1 Element <NameIDPolicy>.....	52
149	3.4.1.2 Element <Scoping>.....	54
150	3.4.1.3 Element <IDPList>.....	54
151	3.4.1.3.1 Element <IDPEntry>.....	55
152	3.4.1.4 Processing Rules.....	55
153	3.4.1.5 Proxying.....	56
154	3.4.1.5.1 Proxying Processing Rules.....	56
155	3.5 Artifact Resolution Protocol.....	58
156	3.5.1 Element <ArtifactResolve>.....	58
157	3.5.2 Element <ArtifactResponse>.....	59
158	3.5.3 Processing Rules.....	59
159	3.6 Name Identifier Management Protocol.....	60
160	3.6.1 Element <ManageNameIDRequest>.....	60
161	3.6.2 Element <ManageNameIDResponse>.....	61
162	3.6.3 Processing Rules.....	61
163	3.7 Single Logout Protocol.....	62
164	3.7.1 Element <LogoutRequest>.....	63
165	3.7.2 Element <LogoutResponse>.....	64
166	3.7.3 Processing Rules.....	64
167	3.7.3.1 Session Participant Rules.....	64
168	3.7.3.2 Session Authority Rules.....	65
169	3.8 Name Identifier Mapping Protocol.....	66
170	3.8.1 Element <NameIDMappingRequest>.....	66
171	3.8.2 Element <NameIDMappingResponse>.....	67
172	3.8.3 Processing Rules.....	67
173	4 SAML Versioning.....	69
174	4.1 SAML Specification Set Version.....	69
175	4.1.1 Schema Version.....	69
176	4.1.2 SAML Assertion Version.....	69

177	4.1.3 SAML Protocol Version.....	70
178	4.1.3.1 Request Version.....	70
179	4.1.3.2 Response Version.....	70
180	4.1.3.3 Permissible Version Combinations.....	71
181	4.2 SAML Namespace Version.....	71
182	4.2.1 Schema Evolution.....	71
183	5 SAML and XML Signature Syntax and Processing.....	72
184	5.1 Signing Assertions.....	72
185	5.2 Request/Response Signing.....	72
186	5.3 Signature Inheritance.....	72
187	5.4 XML Signature Profile.....	73
188	5.4.1 Signing Formats and Algorithms.....	73
189	5.4.2 References.....	73
190	5.4.3 Canonicalization Method.....	73
191	5.4.4 Transforms.....	74
192	5.4.5 KeyInfo.....	74
193	5.4.6 Example.....	74
194	6 SAML and XML Encryption Syntax and Processing.....	78
195	6.1 General Considerations.....	78
196	6.2 Combining Signatures and Encryption[PE43] Key and Data Referencing Guidelines.....	78
197	6.3 Examples.....	79
198	7 SAML Extensibility.....	82
199	7.1 Schema Extension.....	82
200	7.1.1 Assertion Schema Extension.....	82
201	7.1.2 Protocol Schema Extension.....	82
202	7.2 Schema Wildcard Extension Points.....	83
203	7.2.1 Assertion Extension Points.....	83
204	7.2.2 Protocol Extension Points.....	83
205	7.3 Identifier Extension.....	83
206	8 SAML-Defined Identifiers.....	85
207	8.1 Action Namespace Identifiers.....	85
208	8.1.1 Read/Write/Execute/Delete/Control.....	85
209	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	85
210	8.1.3 Get/Head/Put/Post.....	86
211	8.1.4 UNIX File Permissions.....	86
212	8.2 Attribute Name Format Identifiers.....	86
213	8.2.1 Unspecified.....	86
214	8.2.2 URI Reference.....	87
215	8.2.3 Basic.....	87
216	8.3 Name Identifier Format Identifiers.....	87
217	8.3.1 Unspecified.....	87
218	8.3.2 Email Address.....	87
219	8.3.3 X.509 Subject Name.....	87
220	8.3.4 Windows Domain Qualified Name.....	87
221	8.3.5 Kerberos Principal Name.....	88
222	8.3.6 Entity Identifier.....	88
223	8.3.7 Persistent Identifier.....	88
224	8.3.8 Transient Identifier.....	89
225	8.4 Consent Identifiers.....	89

226	8.4.1 Unspecified.....	89
227	8.4.2 Obtained.....	89
228	8.4.3 Prior.....	89
229	8.4.4 Implicit.....	90
230	8.4.5 Explicit.....	90
231	8.4.6 Unavailable.....	90
232	8.4.7 Inapplicable.....	90
233	9 References.....	91
234	9.1 Normative References.....	91
235	9.2 Non-Normative References.....	91
236	Appendix A. Acknowledgments.....	94
237	Appendix B. Notices.....	96
238		

1 Introduction

239

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

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

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

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

1.1 Notation

244

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

246 `Listings of SAML schemas appear like this.`

247

248 `Example code listings appear like this.`

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

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

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

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

253 1.2 Schema Organization and Namespaces

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

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

257 The SAML request-response protocol structures are defined in a schema [SAML-Protocol] associated with
258 the following XML namespace:

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

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

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

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

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

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

268 1.3 Common Data Types

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

271 1.3.1 String Values

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

276 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that
277 have the XML Schema **xs:string** type, or a type derived from that, MUST be compared using an exact
278 binary comparison. In particular, SAML implementations and deployments MUST NOT depend on case-

268 insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific
269 formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft
270 Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

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

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

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

272 1.3.2 URI Values

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

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

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

276 1.3.3 Time Values

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

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

279 1.3.4 ID and ID Reference Values

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

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

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

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

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

2 SAML Assertions

286

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

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

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

- 292 • **Authentication:** The assertion subject was authenticated by a particular means at a particular
293 time.
- 293 • **Attribute:** The assertion subject is associated with the supplied attributes.
- 294 • **Authorization Decision:** A request to allow the assertion subject to access the specified resource
295 has been granted or denied [\[PE13\]or is indeterminate](#).

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

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

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

2.1 Schema Header and Namespace Declarations

303

304 The following schema fragment defines the XML namespaces and other header information for the
305 assertion schema:

```
306 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
307   xmlns="http://www.w3.org/2001/XMLSchema"  
308   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
309   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
310   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  
311   elementFormDefault="unqualified"  
312   attributeFormDefault="unqualified"  
313   blockDefault="substitution"  
314   version="2.0">  
315   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
316     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-  
317     20020212/xmldsig-core-schema.xsd"/>  
318   <import namespace="http://www.w3.org/2001/04/xmlenc#"  
319     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-  
320     20021210/xenc-schema.xsd"/>  
321   <annotation>
```

```

322     <documentation>
323         Document identifier: saml-schema-assertion-2.0
324         Location: http://docs.oasis-open.org/security/saml/v2.0/
325         Revision history:
326         V1.0 (November, 2002):
327             Initial Standard Schema.
328         V1.1 (September, 2003):
329             Updates within the same V1.0 namespace.
330         V2.0 (March, 2005):
331             New assertion schema for SAML V2.0 namespace.
332     </documentation>
333 </annotation>
334 ...
335 </schema>

```

2.2 Name Identifiers

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

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

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

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

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

2.2.1 Element <BaseID>

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

NameQualifier [Optional]

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

SPNameQualifier [Optional]

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

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

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

```
350 <attributeGroup name="IDNameQualifiers">  
351   <attribute name="NameQualifier" type="string" use="optional"/>  
352   <attribute name="SPNameQualifier" type="string" use="optional"/>  
353 </attributeGroup>  
354 <element name="BaseID" type="saml:BaseIDAbstractType"/>  
355 <complexType name="BaseIDAbstractType" abstract="true">  
355   <attributeGroup ref="saml:IDNameQualifiers"/>  
356 </complexType>
```

357 2.2.2 Complex Type `NameIDType`

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

359 `NameQualifier` [Optional]

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

361 `SPNameQualifier` [Optional]

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

363 `Format` [Optional]

364 A URI reference representing the classification of string-based identifier information. See Section
365 8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute
366 and their associated descriptions and processing rules. Unless otherwise specified by an element
367 based on this type, if no `Format` value is provided, then the value
368 `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in
369 effect.

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

366 `SPProvidedID` [Optional]

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

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

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

```
370 <complexType name="NameIDType">
371   <simpleContent>
372     <extension base="string">
373       <attributeGroup ref="saml:IDNameQualifiers"/>
374       <attribute name="Format" type="anyURI" use="optional"/>
375       <attribute name="SPProvidedID" type="string" use="optional"/>
376     </extension>
377   </simpleContent>
378 </complexType>
```

379 2.2.3 Element <NameID>

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

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

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

383 2.2.4 Element <EncryptedID>

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

385 <xenc:EncryptedData> [Required]

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

387 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

```
392 <complexType name="EncryptedElementType">
393   <sequence>
394     <element ref="xenc:EncryptedData"/>
395     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
396   </sequence>
```

```
397 </complexType>
398 <element name="EncryptedID" type="saml:EncryptedElementType"/>
```

399 2.2.5 Element <Issuer>

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

401 Overriding the usual rule for this element's type, if no `Format` value is provided with this element, then
402 the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see Section
403 8.3.6).

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

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

404 2.3 Assertions

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

406 2.3.1 Element <AssertionIDRef>

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

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

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

410 2.3.2 Element <AssertionURIRef>

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

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

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

414 2.3.3 Element <Assertion>

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

416 `Version` [Required]

417 The version of this assertion. The identifier for the version of SAML defined in this specification is
418 "2.0". SAML versioning is discussed in Section 4.

418 `ID` [Required]

419 The identifier for this assertion. It is of type **xs:ID**, and MUST follow the requirements specified in

420 Section 1.3.4 for identifier uniqueness.

421 `IssueInstant` [Required]

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

423 `<Issuer>` [Required]

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

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

426 `<ds:Signature>` [Optional]

427 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as
428 described below and in Section 5.

428 `<Subject>` [Optional]

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

430 `<Conditions>` [Optional]

431 Conditions that MUST be evaluated when assessing the validity of and/or when using the assertion.
432 See Section 2.5 for additional information on how to evaluate conditions.

432 `<Advice>` [Optional]

433 Additional information related to the assertion that assists processing in certain situations but which
434 MAY be ignored by applications that do not understand the advice or do not wish to make use of it.

434 Zero or more of the following statement elements:

435 `<Statement>`

436 A statement of a type defined in an extension schema. An `xsi:type` attribute MUST be used to
437 indicate the actual statement type.

437 `<AuthnStatement>`

438 An authentication statement.

439 `<AuthzDecisionStatement>`

440 An authorization decision statement.

441 `<AttributeStatement>`

442 An attribute statement.

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

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

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

446 If such a signature is used, then the `<ds:Signature>` element MUST be present, and a relying party
447 MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in
448 accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the
449 assertion. If it is valid, then the relying party SHOULD evaluate the signature to determine the identity
450 and appropriateness of the issuer and may continue to process the assertion in accordance with this
451 specification and as it deems appropriate (for example, evaluating conditions, advice, following profile-
452 specific rules, and so on).

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

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

```
449 <element name="Assertion" type="saml:AssertionType"/>
450 <complexType name="AssertionType">
451   <sequence>
452     <element ref="saml:Issuer"/>
453     <element ref="ds:Signature" minOccurs="0"/>
454     <element ref="saml:Subject" minOccurs="0"/>
455     <element ref="saml:Conditions" minOccurs="0"/>
456     <element ref="saml:Advice" minOccurs="0"/>
457     <choice minOccurs="0" maxOccurs="unbounded">
458       <element ref="saml:Statement"/>
459       <element ref="saml:AuthnStatement"/>
460       <element ref="saml:AuthzDecisionStatement"/>
461       <element ref="saml:AttributeStatement"/>
462     </choice>
463   </sequence>
464   <attribute name="Version" type="string" use="required"/>
465   <attribute name="ID" type="ID" use="required"/>
466   <attribute name="IssueInstant" type="dateTime" use="required"/>
467 </complexType>
```

468 2.3.4 Element `<EncryptedAssertion>`

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

470 `<xenc:EncryptedData>` [Required]

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

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

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

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

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

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

477 2.4 Subjects

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

479 2.4.1 Element <Subject>

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

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

482 Identifies the subject.

483 <SubjectConfirmation> [Zero or More]

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

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

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

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

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

```
489 <element name="Subject" type="saml:SubjectType"/>
490 <complexType name="SubjectType">
491   <choice>
492     <sequence>
493       <choice>
494         <element ref="saml:BaseID"/>
495         <element ref="saml:NameID"/>
496         <element ref="saml:EncryptedID"/>
497       </choice>
498       <element ref="saml:SubjectConfirmation" minOccurs="0"
499 maxOccurs="unbounded"/>
500     </sequence>
501     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
502   </choice>
</complexType>
```

503 2.4.1.1 Element <SubjectConfirmation>

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

505 Method [Required]

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

507 <BaseID>, <NameID>, or <EncryptedID> [Optional]
508 Identifies the entity expected to satisfy the enclosing subject confirmation requirements.

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

511 [\[PE47\] If the <SubjectConfirmation> element in an assertion subject contains an identifier the issuer](#)
512 [authorizes the attesting entity to wield the assertion on behalf of that subject. A relying party MAY apply](#)
513 [additional constraints on the use of such an assertion at its discretion, based upon the identities of both](#)
514 [the subject and the attesting entity.](#)

512 [If an assertion is issued for use by an entity other than the subject, then that entity SHOULD be identified](#)
513 [in the <SubjectConfirmation> element.](#)

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

```
514 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
515 <complexType name="SubjectConfirmationType">  
516   <sequence>  
517     <choice minOccurs="0">  
518       <element ref="saml:BaseID"/>  
519       <element ref="saml:NameID"/>  
520       <element ref="saml:EncryptedID"/>  
521     </choice>  
522     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
523   </sequence>  
524   <attribute name="Method" type="anyURI" use="required"/>  
525 </complexType>
```

526 2.4.1.2 Element <SubjectConfirmationData>

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

528 NotBefore [Optional]

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

531 NotOnOrAfter [Optional]

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

534 Recipient [Optional]

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

538 InResponseTo [Optional]

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

542 Address [Optional]

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

544 Arbitrary attributes

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

546 Arbitrary elements

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

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

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

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

```
551 <element name="SubjectConfirmationData"  
552 type="saml:SubjectConfirmationDataType"/>  
552 <complexType name="SubjectConfirmationDataType" mixed="true">  
553   <complexContent>  
554     <restriction base="anyType">  
555       <sequence>  
556         <any namespace="##any" processContents="lax" minOccurs="0"  
557 maxOccurs="unbounded"/>  
557       </sequence>  
558       <attribute name="NotBefore" type="dateTime" use="optional"/>  
559       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
560       <attribute name="Recipient" type="anyURI" use="optional"/>  
561       <attribute name="InResponseTo" type="NCName" use="optional"/>  
562       <attribute name="Address" type="string" use="optional"/>  
563       <anyAttribute namespace="##other" processContents="lax"/>  
564     </restriction>  
565   </complexContent>  
566 </complexType>
```

567 2.4.1.3 Complex Type KeyInfoConfirmationDataType

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

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

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

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

```
572 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
573   <complexContent>  
574     <restriction base="saml:SubjectConfirmationDataType">  
575       <sequence>  
576         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
577       </sequence>  
578     </restriction>  
579   </complexContent>  
580 </complexType>
```

581 2.4.1.4 Example of a Key-Confirmed <Subject>

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

```
583 <Subject>  
584   <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">  
585     scott@example.org  
586   </NameID>  
587   <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-  
588   key">  
589     <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
590       <ds:KeyInfo>  
591         <ds:KeyName>Scott's Key</ds:KeyName>  
592       </ds:KeyInfo>  
593     </SubjectConfirmationData>  
594   </SubjectConfirmation>  
</Subject>
```

595 2.5 Conditions

596 This section defines the SAML constructs that place constraints on the acceptable use of SAML
597 assertions.

597 2.5.1 Element <Conditions>

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

599 NotBefore [Optional]

600 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC,

601 as described in Section 1.3.3.

602 `NotOnOrAfter` [Optional]

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

604 `<Condition>` [Any Number]

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

606 `<AudienceRestriction>` [Any Number]

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

608 `<OneTimeUse>` [Optional]

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

610 `<ProxyRestriction>` [Optional]

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

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

613 The following schema fragment defines the `<Conditions>` element and its **ConditionsType** complex
614 type:

```
614 <element name="Conditions" type="saml:ConditionsType"/>
615 <complexType name="ConditionsType">
616   <choice minOccurs="0" maxOccurs="unbounded">
617     <element ref="saml:Condition"/>
618     <element ref="saml:AudienceRestriction"/>
619     <element ref="saml:OneTimeUse"/>
620     <element ref="saml:ProxyRestriction"/>
621   </choice>
622   <attribute name="NotBefore" type="dateTime" use="optional"/>
623   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
624 </complexType>
```

625 2.5.1.1 General Processing Rules

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

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

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

- 629 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is
630 considered to be **Valid** with respect to condition processing.
- 630 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the
631 assertion is considered to be **Invalid**.
- 631 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, or if an element
632 is encountered that is not understood, then the validity of the assertion cannot be determined and is
633 considered to be **Indeterminate**.
- 632 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then
633 the assertion is considered to be **Valid** with respect to condition processing.

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

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

635 2.5.1.2 Attributes NotBefore and NotOnOrAfter

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

637 The NotBefore attribute specifies the time instant at which the validity interval begins. The
638 NotOnOrAfter attribute specifies the time instant at which the validity interval has ended.

638 If the value for either NotBefore or NotOnOrAfter is omitted, then it is considered unspecified. If the
639 NotBefore attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), then
640 the assertion is **Valid** with respect to conditions at any time before the time instant specified by the
641 NotOnOrAfter attribute. If the NotOnOrAfter attribute is unspecified (and if all other conditions that
642 are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions from the time instant
643 specified by the NotBefore attribute with no expiry. If neither attribute is specified (and if any other
644 conditions that are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions at any
645 time.

639 If both attributes are present, the value for NotBefore MUST be less than (earlier than) the value for
640 NotOnOrAfter.

640 2.5.1.3 Element <Condition>

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

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

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

645 2.5.1.4 Elements <AudienceRestriction> and <Audience>

646 The <AudienceRestriction> element specifies that the assertion is addressed to one or more
647 specific audiences identified by <Audience> elements. Although a SAML relying party that is outside

647 the audiences specified is capable of drawing conclusions from an assertion, the SAML asserting party
648 explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the
649 following element:

648 <Audience>

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

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

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

652 Note that multiple <AudienceRestriction> elements MAY be included in a single assertion, and
653 each MUST be evaluated independently. The effect of this requirement and the preceding definition is
654 that within a given [\[PE46\]<AudienceRestrictions>condition](#), the [<Audience>](#)
655 [elements](#) form a disjunction (an "OR") while multiple [<AudienceRestrictions>](#)
656 [elementconditions](#) form a conjunction (an "AND").

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

```
654 <element name="AudienceRestriction"  
655   type="saml:AudienceRestrictionType"/>  
655 <complexType name="AudienceRestrictionType">  
656   <complexContent>  
657     <extension base="saml:ConditionAbstractType">  
658       <sequence>  
659         <element ref="saml:Audience" maxOccurs="unbounded"/>  
660       </sequence>  
661     </extension>  
662   </complexContent>  
663 </complexType>  
664 <element name="Audience" type="anyURI"/>
```

665 2.5.1.5 Element <OneTimeUse>

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

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

668 The <OneTimeUse> element indicates that the assertion SHOULD be used immediately by the relying
669 party and MUST NOT be retained for future use. Relying parties are always free to request a fresh
670 assertion for every use. However, implementations that choose to retain assertions for future use MUST

669 observe the `<OneTimeUse>` element. This condition is independent from the `NotBefore` and
670 `NotOnOrAfter` condition information.

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

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

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

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

```
674 <element name="OneTimeUse" type="saml:OneTimeUseType"/>  
675 <complexType name="OneTimeUseType">  
676   <complexContent>  
677     <extension base="saml:ConditionAbstractType"/>  
678   </complexContent>  
679 </complexType>
```

680 **2.5.1.6 Element `<ProxyRestriction>`**

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

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

683 `Count` [Optional]

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

685 `<Audience>` [Zero or More]

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

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

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

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

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

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

```
692 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
693 <complexType name="ProxyRestrictionType">
694   <complexContent>
695     <extension base="saml:ConditionAbstractType">
696       <sequence>
697         <element ref="saml:Audience" minOccurs="0"
698 maxOccurs="unbounded"/>
698       </sequence>
699       <attribute name="Count" type="nonNegativeInteger" use="optional"/>
700     </extension>
701   </complexContent>
702 </complexType>
```

703 2.6 Advice

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

705 2.6.1 Element <Advice>

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

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

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

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

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

```
713 <element name="Advice" type="saml:AdviceType"/>
714 <complexType name="AdviceType">
715   <choice minOccurs="0" maxOccurs="unbounded">
716     <element ref="saml:AssertionIDRef"/>
717     <element ref="saml:AssertionURIRef"/>
718     <element ref="saml:Assertion"/>
719     <element ref="saml:EncryptedAssertion"/>
720     <any namespace="##other" processContents="lax"/>
721   </choice>
722 </complexType>
```

723 2.7 Statements

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

725 2.7.1 Element <Statement>

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

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

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

730 2.7.2 Element <AuthnStatement>

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

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

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

734 **AuthnInstant** [Required]

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

736 **SessionIndex** [Optional]

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

738 **SessionNotOnOrAfter** [Optional]

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

740 <SubjectLocality> [Optional]

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

742 <AuthnContext> [Required]

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

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

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

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

```
748 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
749 <complexType name="AuthnStatementType">
750   <complexContent>
751     <extension base="saml:StatementAbstractType">
752       <sequence>
753         <element ref="saml:SubjectLocality" minOccurs="0"/>
754         <element ref="saml:AuthnContext"/>
755       </sequence>
756       <attribute name="AuthnInstant" type="dateTime" use="required"/>
757       <attribute name="SessionIndex" type="string" use="optional"/>
758       <attribute name="SessionNotOnOrAfter" type="dateTime"
759 use="optional"/>
759     </extension>
760   </complexContent>
761 </complexType>
```

762 2.7.2.1 Element `<SubjectLocality>`

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

764 Address [Optional]

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

766 DNSName [Optional]

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

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

769 The following schema fragment defines the `<SubjectLocality>` element and its
770 **SubjectLocalityType** complex type:

```
770 <element name="SubjectLocality" type="saml:SubjectLocalityType"/>
771 <complexType name="SubjectLocalityType">
772   <attribute name="Address" type="string" use="optional"/>
773   <attribute name="DNSName" type="string" use="optional"/>
774 </complexType>
```

775 2.7.2.2 Element `<AuthnContext>`

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

777 <AuthnContextClassRef> [Optional]
778 A URI reference identifying an authentication context class that describes the authentication context
779 declaration that follows.

779 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]
780 Either an authentication context declaration provided by value, or a URI reference that identifies
781 such a declaration. The URI reference MAY directly resolve into an XML document containing the
782 referenced declaration.

781 <AuthenticatingAuthority> [Zero or More]
782 Zero or more unique identifiers of authentication authorities that were involved in the authentication
783 of the principal (not including the assertion issuer, who is presumed to have been involved without
784 being explicitly named here).

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

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

```
785 <element name="AuthnContext" type="saml:AuthnContextType"/>  
786 <complexType name="AuthnContextType">  
787   <sequence>  
788     <choice>  
789       <sequence>  
790         <element ref="saml:AuthnContextClassRef"/>  
791         <choice minOccurs="0">  
792           <element ref="saml:AuthnContextDecl"/>  
793           <element ref="saml:AuthnContextDeclRef"/>  
794         </choice>  
795       </sequence>  
796     <choice>  
797       <element ref="saml:AuthnContextDecl"/>  
798       <element ref="saml:AuthnContextDeclRef"/>  
799     </choice>  
800   </choice>  
801   <element ref="saml:AuthenticatingAuthority" minOccurs="0"  
802   maxOccurs="unbounded"/>  
802 </sequence>  
803 </complexType>  
804 <element name="AuthnContextClassRef" type="anyURI"/>  
805 <element name="AuthnContextDeclRef" type="anyURI"/>  
806 <element name="AuthnContextDecl" type="anyType"/>  
807 <element name="AuthenticatingAuthority" type="anyURI"/>
```

808 2.7.3 Element <AttributeStatement>

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

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

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

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

813 The following schema fragment defines the `<AttributeStatement>` element and its
814 **AttributeStatementType** complex type:

```
814 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
815 <complexType name="AttributeStatementType">
816   <complexContent>
817     <extension base="saml:StatementAbstractType">
818       <choice maxOccurs="unbounded">
819         <element ref="saml:Attribute"/>
820         <element ref="saml:EncryptedAttribute"/>
821       </choice>
822     </extension>
823   </complexContent>
824 </complexType>
```

825 2.7.3.1 Element `<Attribute>`

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

827 Name [Required]

828 The name of the attribute.

829 NameFormat [Optional]

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

831 FriendlyName [Optional]

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

833 Arbitrary attributes

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

835 `<AttributeValue>` [Any Number]

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

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

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

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

```
840 <element name="Attribute" type="saml:AttributeType"/>
841 <complexType name="AttributeType">
842   <sequence>
843     <element ref="saml:AttributeValue" minOccurs="0"
844 maxOccurs="unbounded"/>
844   </sequence>
845   <attribute name="Name" type="string" use="required"/>
846   <attribute name="NameFormat" type="anyURI" use="optional"/>
847   <attribute name="FriendlyName" type="string" use="optional"/>
848   <anyAttribute namespace="##other" processContents="lax"/>
849 </complexType>
```

850 2.7.3.1.1 Element `<AttributeValue>`

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

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

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

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

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

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

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

858 2.7.3.2 Element `<EncryptedAttribute>`

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

860 `<xenc:EncryptedData>` [Required]

861 The encrypted content and associated encryption details, as defined by the XML Encryption
862 Syntax and Processing specification [XMLEnc]. The `Type` attribute **SHOULD** be present and, if

862 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
863 encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

863 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

868 2.7.4 Element <AuthzDecisionStatement>

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

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

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

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

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

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

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

- 877 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case
878 insensitive, a requester SHOULD NOT be able to gain access to a denied resource by changing
879 the case of a part of the resource URI reference.

- 878 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
879 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
880 gain access to a denied resource by creating such an equivalence.

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

880 Resource [Required]

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

882 Decision [Required]

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

884 `<Action>` [One or more]

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

886 `<Evidence>` [Optional]

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

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

```
889 <element name="AuthzDecisionStatement"  
890 type="saml:AuthzDecisionStatementType"/>  
890 <complexType name="AuthzDecisionStatementType">  
891   <complexContent>  
892     <extension base="saml:StatementAbstractType">  
893       <sequence>  
894         <element ref="saml:Action" maxOccurs="unbounded"/>  
895         <element ref="saml:Evidence" minOccurs="0"/>  
896       </sequence>  
897       <attribute name="Resource" type="anyURI" use="required"/>  
898       <attribute name="Decision" type="saml:DecisionType"  
899 use="required"/>  
899     </extension>  
900   </complexContent>  
901 </complexType>
```

902 2.7.4.1 Simple Type DecisionType

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

904 Permit

905 The specified action is permitted.

906 Deny

907 The specified action is denied.

908 Indeterminate

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

910 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability
911 to provide an affirmative statement but where it is not able to issue a decision. Additional information as

911 to the reason for the refusal or inability to provide a decision MAY be returned as <StatusDetail>
912 elements in the enclosing <Response>.

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

```
913 <simpleType name="DecisionType">  
914   <restriction base="string">  
915     <enumeration value="Permit"/>  
916     <enumeration value="Deny"/>  
917     <enumeration value="Indeterminate"/>  
918   </restriction>  
919 </simpleType>
```

920 **2.7.4.2 Element <Action>**

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

922 Namespace [[PE36](#)]RequiredOptional]

923 A URI reference representing the namespace in which the name of the specified action is to be
924 interpreted. ~~If this element is absent, the namespace-~~
925 ~~urn:oasis:names:tc:SAML:1.0:action:rwede negation specified in Section 8.1.2 is in~~
926 ~~effect.~~

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

```
925 <element name="Action" type="saml:ActionType"/>  
926 <complexType name="ActionType">  
927   <simpleContent>  
928     <extension base="string">  
929       <attribute name="Namespace" type="anyURI" use="required"/>  
930     </extension>  
931   </simpleContent>  
932 </complexType>
```

933 **2.7.4.3 Element <Evidence>**

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

935 <AssertionIDRef> [Any number]

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

937 <AssertionURIRef> [Any number]

938 Specifies an assertion by means of a URI reference.

939 <Assertion> [Any number]

940 Specifies an assertion by value.

941 <EncryptedAssertion> [Any number]

942 Specifies an encrypted assertion by value.

943 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
944 and the SAML authority making the authorization decision. For example, in the case that the SAML
945 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use

944 that assertion as evidence in making its authorization decision without endorsing the <Evidence>
945 element's assertion as valid either to the relying party or any other third party.

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

```
946 <element name="Evidence" type="saml:EvidenceType"/>  
947 <complexType name="EvidenceType">  
948   <choice maxOccurs="unbounded">  
949     <element ref="saml:AssertionIDRef"/>  
950     <element ref="saml:AssertionURIRef"/>  
951     <element ref="saml:Assertion"/>  
952     <element ref="saml:EncryptedAssertion"/>  
953   </choice>  
954 </complexType>
```

3 SAML Protocols

955

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

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

958



960

Figure 1: SAML Request-Response Protocol

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

962 The protocols defined by SAML achieve the following actions:

- 963 • Returning one or more requested assertions. This can occur in response to either a direct request
964 for specific assertions or a query for assertions that meet particular criteria.
- 965 • Performing authentication on request and returning the corresponding assertion
- 966 • Registering a name identifier or terminating a name registration on request
- 967 • Retrieving a protocol message that has been requested by means of an artifact
- 968 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
969 request
- 970 • Providing a name identifier mapping on request

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

3.1 Schema Header and Namespace Declarations

972

973 The following schema fragment defines the XML namespaces and other header information for the
974 protocol schema:

```
974 <schema  
975   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
976   xmlns="http://www.w3.org/2001/XMLSchema"  
977   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
978   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
979   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
980   elementFormDefault="unqualified"  
981   attributeFormDefault="unqualified"
```

```

982     blockDefault="substitution"
983     version="2.0">
984     <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
985           schemaLocation="saml-schema-assertion-2.0.xsd"/>
986     <import namespace="http://www.w3.org/2000/09/xmlsig#"
987           schemaLocation="http://www.w3.org/TR/2002/REC-xmlsig-core-
988 20020212/xmlsig-core-schema.xsd"/>
989     <annotation>
990       <documentation>
991         Document identifier: saml-schema-protocol-2.0
992         Location: http://docs.oasis-open.org/security/saml/v2.0/
993         Revision history:
994         V1.0 (November, 2002):
995           Initial Standard Schema.
996         V1.1 (September, 2003):
997           Updates within the same V1.0 namespace.
998         V2.0 (March, 2005):
999           New protocol schema based in a SAML V2.0 namespace.
1000       </documentation>
1001     </annotation>
1002 ...
1003 </schema>

```

1003 3.2 Requests and Responses

1004 The following sections define the SAML constructs and basic requirements that underlie all of the request
1005 and response messages used in SAML protocols.

1005 3.2.1 Complex Type RequestAbstractType

1006 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1007 This type defines common attributes and elements that are associated with all SAML requests:

1007 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**
1008 for V2.0 of SAML.

1008 ID [Required]

1009 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in
1010 Section 1.3.4 for identifier uniqueness. The values of the **ID** attribute in a request and the
1011 **InResponseTo** attribute in the corresponding response MUST match.

1010 Version [Required]

1011 The version of this request. The identifier for the version of SAML defined in this specification is
1012 "2.0". SAML versioning is discussed in Section 4.

1012 IssueInstant [Required]

1013 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
1014 1.3.3.

1014 Destination [Optional]

1015 A URI reference indicating the address to which this request has been sent. This is useful to prevent
1016 malicious forwarding of requests to unintended recipients, a protection that is required by some
1017 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies
1018 the location at which the message was received. If it does not, the request MUST be discarded.
1019 Some protocol bindings may require the use of this attribute (see [SAMLBind]).

1016 Consent [Optional]

1017 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1018 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value
1019 of the Consent attribute and their associated descriptions. If no Consent value is provided, the
1020 identifier urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in
1021 effect.

1018 <saml:Issuer> [Optional]

1019 Identifies the entity that generated the request message. (For more information on this element, see
1020 Section 2.2.5.)

1020 <ds:Signature> [Optional]

1021 An XML Signature that authenticates the requester and provides message integrity, as described
1022 below and in Section 5.

1022 <Extensions> [Optional]

1023 This extension point contains optional protocol message extension elements that are agreed on
1024 between the communicating parties. No extension schema is required in order to make use of this
1025 extension point, and even if one is provided, the lax validation setting does not impose a
1026 requirement for the extension to be valid. SAML extension elements MUST be namespace-qualified
1027 in a non-SAML-defined namespace.

1024 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to
1025 authenticate itself, and message integrity may often be required. Authentication and message integrity
1026 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request
1027 MAY be signed, which provides both authentication of the requester and message integrity.

1025 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1026 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)
1027 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the
1028 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the
1029 signature to determine the identity and appropriateness of the signer and may continue to process the
1030 request or respond with an error (if the request is invalid for some other reason).

1026 If a Consent attribute is included and the value indicates that some form of principal consent has been
1027 obtained, then the request SHOULD be signed.

1027 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if
1028 it responds, it MUST return a SAML response message with a <StatusCode> element with the value
1029 urn:oasis:names:tc:SAML:2.0:status:Requester. In some cases, for example during a
1030 suspected denial-of-service attack, not responding at all may be warranted.

1028 The following schema fragment defines the **RequestAbstractType** complex type:

```
1029 <complexType name="RequestAbstractType" abstract="true">  
1030   <sequence>  
1031     <element ref="saml:Issuer" minOccurs="0"/>  
1032     <element ref="ds:Signature" minOccurs="0"/>  
1033     <element ref="samlp:Extensions" minOccurs="0"/>  
1034   </sequence>  
1035   <attribute name="ID" type="ID" use="required"/>  
1036   <attribute name="Version" type="string" use="required"/>  
1037   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1038   <attribute name="Destination" type="anyURI" use="optional"/>  
1039   <attribute name="Consent" type="anyURI" use="optional"/>  
1040 </complexType>  
1041 <element name="Extensions" type="samlp:ExtensionsType"/>  
1042 <complexType name="ExtensionsType">  
1043   <sequence>
```

1044
1045
1046

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

1047 3.2.2 Complex Type StatusResponseType

1048 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This
1049 type defines common attributes and elements that are associated with all SAML responses:

1049 ID [Required]

1050 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in
1051 Section 1.3.4 for identifier uniqueness.

1051 InResponseTo [Optional]

1052 A reference to the identifier of the request to which the response corresponds, if any. If the response
1053 is not generated in response to a request, or if the ID attribute value of a request cannot be
1054 determined (for example, the request is malformed), then this attribute MUST NOT be present.
1055 Otherwise, it MUST be present and its value MUST match the value of the corresponding request's
1056 ID attribute.

1053 Version [Required]

1054 The version of this response. The identifier for the version of SAML defined in this specification is
1055 "2.0". SAML versioning is discussed in Section 4.

1055 IssueInstant [Required]

1056 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
1057 1.3.3.

1057 Destination [Optional]

1058 A URI reference indicating the address to which this response has been sent. This is useful to
1059 prevent malicious forwarding of responses to unintended recipients, a protection that is required by
1060 some protocol bindings. If it is present, the actual recipient MUST check that the URI reference
1061 identifies the location at which the message was received. If it does not, the response MUST be
1062 discarded. Some protocol bindings may require the use of this attribute (see [SAMLBind]).

1059 Consent [Optional]

1060 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1061 the sending of this response. See Section 8.4 for some URI references that MAY be used as the
1062 value of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided,
1063 the identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in
1064 effect.

1061 <saml:Issuer> [Optional]

1062 Identifies the entity that generated the response message. (For more information on this element,
1063 see Section 2.2.5.)

1063 <ds:Signature> [Optional]

1064 An XML Signature that authenticates the responder and provides message integrity, as described
1065 below and in Section 5.

1065 <Extensions> [Optional]

1066 This extension point contains optional protocol message extension elements that are agreed on
1067 between the communicating parties. . No extension schema is required in order to make use of this
1068 extension point, and even if one is provided, the lax validation setting does not impose a

1067 requirement for the extension to be valid. SAML extension elements MUST be namespace-qualified
1068 in a non-SAML-defined namespace.

1068 <Status> [Required]

1069 A code representing the status of the corresponding request.

1070 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to
1071 authenticate itself, and message integrity may often be required. Authentication and message integrity
1072 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML
1073 response MAY be signed, which provides both authentication of the responder and message integrity.

1071 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1072 requester receiving the response MUST verify that the signature is valid (that is, that the message has
1073 not been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely
1074 on the contents of the response and SHOULD treat it as an error. If it is valid, then the requester
1075 SHOULD evaluate the signature to determine the identity and appropriateness of the signer and may
1076 continue to process the response as it deems appropriate.

1072 If a Consent attribute is included and the value indicates that some form of principal consent has been
1073 obtained, then the response SHOULD be signed.

1073 The following schema fragment defines the **StatusResponseType** complex type:

```
1074 <complexType name="StatusResponseType">  
1075   <sequence>  
1076     <element ref="saml:Issuer" minOccurs="0"/>  
1077     <element ref="ds:Signature" minOccurs="0"/>  
1078     <element ref="samlp:Extensions" minOccurs="0"/>  
1079     <element ref="samlp:Status"/>  
1080   </sequence>  
1081   <attribute name="ID" type="ID" use="required"/>  
1082   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1083   <attribute name="Version" type="string" use="required"/>  
1084   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1085   <attribute name="Destination" type="anyURI" use="optional"/>  
1086   <attribute name="Consent" type="anyURI" use="optional"/>  
1087 </complexType>
```

1088 3.2.2.1 Element <Status>

1089 The <Status> element contains the following elements:

1090 <StatusCode> [Required]

1091 A code representing the status of the activity carried out in response to the corresponding request.

1092 <StatusMessage> [Optional]

1093 A message which MAY be returned to an operator.

1094 <StatusDetail> [Optional]

1095 Additional information concerning the status of the request.

1096 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1097 <element name="Status" type="samlp:StatusType"/>  
1098 <complexType name="StatusType">  
1099   <sequence>  
1100     <element ref="samlp:StatusCode"/>  
1101     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1102     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1103   </sequence>
```


1104 </complexType>

1105 3.2.2.2 Element <StatusCode>

1106 The <StatusCode> element specifies a code or a set of nested codes representing the status of the
1107 corresponding request. The <StatusCode> element has the following element and attribute:

1107 Value [Required]

1108 The status code value. This attribute contains a URI reference. The value of the topmost
1109 <StatusCode> element MUST be from the top-level list provided in this section.

1109 <StatusCode> [Optional]

1110 A subordinate status code that provides more specific information on an error condition. Note that
1111 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for
1112 additional information by intentionally presenting erroneous requests.

1111 The permissible top-level <StatusCode> values are as follows:

1112 urn:oasis:names:tc:SAML:2.0:status:Success

1113 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1114 <StatusDetail> elements.

1114 urn:oasis:names:tc:SAML:2.0:status:Requester

1115 The request could not be performed due to an error on the part of the requester.

1116 urn:oasis:names:tc:SAML:2.0:status:Responder

1117 The request could not be performed due to an error on the part of the SAML responder or SAML
1118 authority.

1118 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1119 The SAML responder could not process the request because the version of the request message was
1120 incorrect.

1120 The following second-level status codes are referenced at various places in this specification. Additional
1121 second-level status codes MAY be defined in future versions of the SAML specification. System entities
1122 are free to define more specific status codes by defining appropriate URI references.

1121 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1122 The responding provider was unable to successfully authenticate the principal.

1123 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1124 Unexpected or invalid content was encountered within a <saml:Attribute> or
1125 <saml:AttributeValue> element.

1125 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1126 The responding provider cannot or will not support the requested name identifier policy.

1127 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1128 The specified authentication context requirements cannot be met by the responder.

1129 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1130 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in
1131 an <IDPList> can be resolved or that none of the supported identity providers are available.

1131 urn:oasis:names:tc:SAML:2.0:status:NoPassive
1132 Indicates the responding provider cannot authenticate the principal passively, as has been
1133 requested.

1134 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
1135 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1136 supported by the intermediary.

1137 urn:oasis:names:tc:SAML:2.0:status:PartialLogout
1138 Used by a session authority to indicate to a session participant that it was not able to propagate
1139 logout to all other session participants.

1140 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
1141 Indicates that a responding provider cannot authenticate the principal directly and is not permitted to
1142 proxy the request further.

1143 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
1144 The SAML responder or SAML authority is able to process the request but has chosen not to
1145 respond. This status code MAY be used when there is concern about the security context of the
1146 request message or the sequence of request messages received from a particular requester.

1147 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
1148 The SAML responder or SAML authority does not support the request.

1149 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
1150 The SAML responder cannot process any requests with the protocol version specified in the request.

1151 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
1152 The SAML responder cannot process the request because the protocol version specified in the
1153 request message is a major upgrade from the highest protocol version supported by the responder.

1154 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
1155 The SAML responder cannot process the request because the protocol version specified in the
1156 request message is too low.

1157 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
1158 The resource value provided in the request message is invalid or unrecognized.

1159 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
1160 The response message would contain more elements than the SAML responder is able to return.

1161 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1162 An entity that has no knowledge of a particular attribute profile has been presented with an attribute
1163 drawn from that profile.

1164 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
1165 The responding provider does not recognize the principal specified or implied by the request.

1166 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
1167 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
1168 request.

1159 The following schema fragment defines the `<StatusCode>` element and its **StatusCodeType** complex
1160 type:

```
1160 <element name="StatusCode" type="samlp:StatusCodeType"/>
1161 <complexType name="StatusCodeType">
1162   <sequence>
1163     <element ref="samlp:StatusCode" minOccurs="0"/>
1164   </sequence>
1165   <attribute name="Value" type="anyURI" use="required"/>
1166 </complexType>
```

1167 3.2.2.3 Element `<StatusMessage>`

1168 The `<StatusMessage>` element specifies a message that MAY be returned to an operator:

1169 The following schema fragment defines the `<StatusMessage>` element:

```
1170 <element name="StatusMessage" type="string"/>
```

1171 3.2.2.4 Element `<StatusDetail>`

1172 The `<StatusDetail>` element MAY be used to specify additional information concerning the status of
1173 the request. The additional information consists of zero or more elements from any namespace, with no
1174 requirement for a schema to be present or for schema validation of the `<StatusDetail>` contents.

1173 The following schema fragment defines the `<StatusDetail>` element and its **StatusDetailType**
1174 complex type:

```
1174 <element name="StatusDetail" type="samlp:StatusDetailType"/>
1175 <complexType name="StatusDetailType">
1176   <sequence>
1177     <any namespace="##any" processContents="lax" minOccurs="0"
1178     maxOccurs="unbounded"/>
1178   </sequence>
1179 </complexType>
```

1180 3.3 Assertion Query and Request Protocol

1181 This section defines messages and processing rules for requesting existing assertions by reference or
1182 querying for assertions by subject and statement type.

1182 3.3.1 Element `<AssertionIDRequest>`

1183 If the requester knows the unique identifier of one or more assertions, the `<AssertionIDRequest>`
1184 message element can be used to request that they be returned in a `<Response>` message. The
1185 `<saml:AssertionIDRef>` element is used to specify each assertion to return. See Section 2.3.1 for
1186 more information on this element.

1184 The following schema fragment defines the `<AssertionIDRequest>` element:

```
1185 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
1186 <complexType name="AssertionIDRequestType">
1187   <complexContent>
1188     <extension base="samlp:RequestAbstractType">
1189       <sequence>
1190         <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>
1191       </sequence>
1192     </extension>
1193   </complexContent>
```

1194 </complexType>

1195 3.3.2 Queries

1196 The following sections define the SAML query request messages.

1197 3.3.2.1 Element <SubjectQuery>

1198 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1199 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract
1200 and is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the
1201 <saml:Subject> element (defined in Section 2.4) to **RequestAbstractType**.

1199 The following schema fragment defines the <SubjectQuery> element and its
1200 **SubjectQueryAbstractType** complex type:

```
1200 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
1201 <complexType name="SubjectQueryAbstractType" abstract="true">
1202   <complexContent>
1203     <extension base="samlp:RequestAbstractType">
1204       <sequence>
1205         <element ref="saml:Subject"/>
1206       </sequence>
1207     </extension>
1208   </complexContent>
1209 </complexType>
```

1210 3.3.2.2 Element <AuthnQuery>

1211 The <AuthnQuery> message element is used to make the query “What assertions containing
1212 authentication statements are available for this subject?” A successful <Response> will contain one or
1213 more assertions containing authentication statements.

1212 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1213 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1214 that have occurred in a previous interaction between the indicated subject and the authentication
1215 authority.

1213 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition
1214 of the following element and attribute:

1214 SessionIndex [Optional]

1215 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1216 containing authentication statements do you have for this subject within the context of the supplied
1217 session information?”

1216 <RequestedAuthnContext> [Optional]

1217 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1218 containing authentication statements do you have for this subject that satisfy the authentication
1219 context requirements in this element?"

1218 In response to an authentication query, a SAML authority returns assertions with authentication
1219 statements as follows:

- 1219 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1220 assertions that may be returned.

- 1220 • If the `SessionIndex` attribute is present in the query, at least one `<AuthnStatement>` element
1221 in the set of returned assertions MUST contain a `SessionIndex` attribute that matches the
1222 `SessionIndex` attribute in the query. It is OPTIONAL for the complete set of all such matching
1223 assertions to be returned in the response.
- 1221 • If the `<RequestedAuthnContext>` element is present in the query, at least one
1222 `<AuthnStatement>` element in the set of returned assertions MUST contain an
1223 `<AuthnContext>` element that satisfies the element in the query (see Section 3.3.2.2.1). It is
1224 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1222 The following schema fragment defines the `<AuthnQuery>` element and its **AuthnQueryType** complex
1223 type:

```
1223 <element name="AuthnQuery" type="saml:AuthnQueryType"/>
1224 <complexType name="AuthnQueryType">
1225   <complexContent>
1226     <extension base="saml:SubjectQueryAbstractType">
1227       <sequence>
1228         <element ref="saml:RequestedAuthnContext" minOccurs="0"/>
1229       </sequence>
1230       <attribute name="SessionIndex" type="string" use="optional"/>
1231     </extension>
1232   </complexContent>
1233 </complexType>
```

1234 3.3.2.2.1 Element `<RequestedAuthnContext>`

1235 The `<RequestedAuthnContext>` element specifies the authentication context requirements of
1236 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
1237 complex type defines the following elements and attributes:

1236 `<saml:AuthnContextClassRef>` or `<saml:AuthnContextDeclRef>` [One or More]

1237 Specifies one or more URI references identifying authentication context classes or declarations.
1238 These elements are defined in Section 2.7.2.2. For more information about authentication context
1239 classes, see [SAMLAuthnCxt].

1238 Comparison [Optional]

1239 Specifies the comparison method used to evaluate the requested context classes or statements, one
1240 of "exact", "minimum", "maximum", or "better". The default is "exact".

1240 Either a set of class references or a set of declaration references can be used. [\[PE45\]f ordering is](#)
1241 [relevant to the evaluation of the request, then the set of supplied references MUST be evaluated as an](#)
1242 [ordered set, where the first element is the most preferred authentication context class or declaration. For](#)
1243 [example, ordering is significant when using this element in an `<AuthnRequest>` message but not in an](#)
1244 [`<AuthnQuery>` message.](#)

1241 If none of the specified classes or declarations can be satisfied in accordance with the rules below, then
1242 the responder MUST return a `<Response>` message with a second-level `<StatusCode>` of
1243 `urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext`.

1242 If `Comparison` is set to "exact" or omitted, then the resulting authentication context in the
1243 authentication statement MUST be the exact match of at least one of the authentication contexts
1244 specified.

1243 If `Comparison` is set to "minimum", then the resulting authentication context in the authentication
1244 statement MUST be at least as strong (as deemed by the responder) as one of the authentication
1245 contexts specified.

1244 If `Comparison` is set to "better", then the resulting authentication context in the authentication
1245 statement MUST be stronger (as deemed by the responder) than [\[PE45\]](#)any one of the authentication
1246 contexts specified.

1245 If `Comparison` is set to "maximum", then the resulting authentication context in the authentication
1246 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength
1247 of at least one of the authentication contexts specified.

1246 The following schema fragment defines the `<RequestedAuthnContext>` element and its
1247 **RequestedAuthnContextType** complex type:

```
1247 <element name="RequestedAuthnContext"  
1248 type="samlp:RequestedAuthnContextType"/>  
1248 <complexType name="RequestedAuthnContextType">  
1249   <choice>  
1250     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>  
1251     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>  
1252   </choice>  
1253   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"  
1254 use="optional"/>  
1254 </complexType>  
1255 <simpleType name="AuthnContextComparisonType">  
1256   <restriction base="string">  
1257     <enumeration value="exact"/>  
1258     <enumeration value="minimum"/>  
1259     <enumeration value="maximum"/>  
1260     <enumeration value="better"/>  
1261   </restriction>  
1262 </simpleType>
```

1263 3.3.2.3 Element `<AttributeQuery>`

1264 The `<AttributeQuery>` element is used to make the query "Return the requested attributes for this
1265 subject." A successful response will be in the form of assertions containing attribute statements, to the
1266 extent allowed by policy. This element is of type **AttributeQueryType**, which extends
1267 **SubjectQueryAbstractType** with the addition of the following element:

1265 `<saml:Attribute>` [Any Number]

1266 Each `<saml:Attribute>` element specifies an attribute whose value(s) are to be returned. If no
1267 attributes are specified, it indicates that all attributes allowed by policy are requested. If a given
1268 `<saml:Attribute>` element contains one or more `<saml:AttributeValue>` elements, then if
1269 that attribute is returned in the response, it MUST NOT contain any values that are not equal to the
1270 values specified in the query. In the absence of equality rules specified by particular profiles or
1271 attributes, equality is defined as an identical XML representation of the value. For more information
1272 on `<saml:Attribute>`, see Section 2.7.3.1.

1267 A single query MUST NOT contain two `<saml:Attribute>` elements with the same `Name` and
1268 `NameFormat` values (that is, a given attribute MUST be named only once in a query).

1268 In response to an attribute query, a SAML authority returns assertions with attribute statements as
1269 follows:

- 1269 • Rules given in Section 3.3.4 for matching against the `<Subject>` element of the query identify the
1270 assertions that may be returned.
- 1270 • If any `<Attribute>` elements are present in the query, they constrain/filter the attributes and
1271 optionally the values returned, as noted above.
- 1271 • The attributes and values returned MAY also be constrained by application-specific policy
1272 considerations.

1272 The second-level status codes `urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile`
1273 and `urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue` MAY be used to
1274 indicate problems with the interpretation of attribute or value information in a query.

1273 The following schema fragment defines the `<AttributeQuery>` element and its **AttributeQueryType**
1274 complex type:

```
1274 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1275 <complexType name="AttributeQueryType">
1276   <complexContent>
1277     <extension base="samlp:SubjectQueryAbstractType">
1278       <sequence>
1279         <element ref="saml:Attribute" minOccurs="0"
1280 maxOccurs="unbounded"/>
1280       </sequence>
1281     </extension>
1282   </complexContent>
1283 </complexType>
```

1284 3.3.2.4 Element `<AuthzDecisionQuery>`

1285 The `<AuthzDecisionQuery>` element is used to make the query “Should these actions on this
1286 resource be allowed for this subject, given this evidence?” A successful response will be in the form of
1287 assertions containing authorization decision statements.

1286 **Note:** The `<AuthzDecisionQuery>` feature has been frozen as of SAML V2.0, with no
1287 future enhancements planned. Users who require additional functionality may want to
1288 consider the eXtensible Access Control Markup Language [XACML], which offers
1289 enhanced authorization decision features.

1287 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1288 addition of the following elements and attribute:

1288 Resource [Required]

1289 A URI reference indicating the resource for which authorization is requested.

1290 `<saml:Action>` [One or More]

1291 The actions for which authorization is requested. For more information on this element, see Section
1292 2.7.4.2.

1292 `<saml:Evidence>` [Optional]

1293 A set of assertions that the SAML authority MAY rely on in making its authorization decision. For
1294 more information on this element, see Section 2.7.4.3.

1294 In response to an authorization decision query, a SAML authority returns assertions with authorization
1295 decision statements as follows:

- 1295 • Rules given in Section 3.3.4 for matching against the `<Subject>` element of the query identify the
1296 assertions that may be returned.

1296 The following schema fragment defines the `<AuthzDecisionQuery>` element and its
1297 **AuthzDecisionQueryType** complex type:

```
1297 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
1298 <complexType name="AuthzDecisionQueryType">
1299   <complexContent>
1300     <extension base="samlp:SubjectQueryAbstractType">
1301       <sequence>
1302         <element ref="saml:Action" maxOccurs="unbounded"/>

```

```

1303         <element ref="saml:Evidence" minOccurs="0"/>
1304     </sequence>
1305     <attribute name="Resource" type="anyURI" use="required"/>
1306 </extension>
1307 </complexContent>
1308 </complexType>

```

1309 3.3.3 Element <Response>

1310 The <Response> message element is used when a response consists of a list of zero or more
 1311 assertions that satisfy the request. It has the complex type **ResponseType**, which extends
 1312 **StatusResponseType** and adds the following elements:

1311 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1312 Specifies an assertion by value, or optionally an encrypted assertion by value. See Section 2.3.3 for
 1313 more information on these elements.

1313 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1314 <element name="Response" type="samlp:ResponseType"/>
1315 <complexType name="ResponseType">
1316   <complexContent>
1317     <extension base="samlp:StatusResponseType">
1318       <choice minOccurs="0" maxOccurs="unbounded">
1319         <element ref="saml:Assertion"/>
1320         <element ref="saml:EncryptedAssertion"/>
1321       </choice>
1322     </extension>
1323   </complexContent>
1324 </complexType>

```

1325 3.3.4 Processing Rules

1326 In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST**
 1327 contain a <saml:Subject> element that **strongly matches** the <saml:Subject> element found in
 1328 the query.

1327 A <saml:Subject> element S1 strongly matches S2 if and only if the following two conditions both
 1328 apply:

- 1328 • If S2 includes an identifier element (<BaseID>, <NameID>, or <EncryptedID>), then S1 **MUST**
 1329 include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or
 1330 S2. In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical"
 1331 means that the identifier element's content and attribute values **MUST** be the same. An encrypted
 1332 identifier will be identical to the original according to this definition, once decrypted.
- 1329 • If S2 includes one or more <saml:SubjectConfirmation> elements, then S1 **MUST** include at
 1330 least one <saml:SubjectConfirmation> element such that S1 can be confirmed in the manner
 1331 described by at least one <saml:SubjectConfirmation> element in S2.

1330 As an example of what is and is not permitted, S1 could contain a <saml:NameID> with a particular
 1331 Format value, and S2 could contain a <saml:EncryptedID> element that is the result of encrypting
 1332 S1's <saml:NameID> element. However, S1 and S2 cannot contain a <saml:NameID> element with
 1333 different Format values and element content, even if the two identifiers are considered to refer to the
 1334 same principal.

1331 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
 1332 expressed by a query or assertion reference, the <Response> element **MUST NOT** contain an

1332 <Assertion> element and MUST include a <StatusCode> element with the value
1333 urn:oasis:names:tc:SAML:2.0:status:Success.

1333 All other processing rules associated with the underlying request and response messages MUST be
1334 observed.

1334 3.4 Authentication Request Protocol

1335 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
1336 authentication statements to establish a security context at one or more relying parties, it can use the
1337 authentication request protocol to send an <AuthnRequest> message element to a SAML authority and
1338 request that it return a <Response> message containing one or more such assertions. Such assertions
1339 MAY contain additional statements of any type, but at least one assertion MUST contain at least one
1340 authentication statement. A SAML authority that supports this protocol is also termed an identity
1341 provider.

1336 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
1337 context of use. Also, the exact means by which the principal or agent authenticates to the identity
1338 provider is not specified, though the means of authentication might impact the content of the response.
1339 Other issues related to the validation of authentication credentials by the identity provider or any
1340 communication between the identity provider and any other entities involved in the authentication
1341 process are also out of scope of this protocol.

1337 The descriptions and processing rules in the following sections reference the following actors, many of
1338 whom might be the same entity in a particular profile of use:

1338 Requester

1339 The entity who creates the authentication request and to whom the response is to be returned.

1340 Presenter

1341 The entity who presents the request to the identity provider and either authenticates itself during
1342 the transmission of the message, or relies on an existing security context to establish its identity.
1343 If not the requester, the presenter acts as an intermediary between the requester and the
1344 responding identity provider.

1342 Requested Subject

1343 The entity about whom one or more assertions are being requested.

1344 Attesting Entity

1345 The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>
1346 elements of the resulting assertion(s).

1346 Relying Party

1347 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
1348 the profile or context of use, generally to establish a security context.

1348 Identity Provider

1349 The entity to whom the presenter gives the request and from whom the presenter receives the
1350 response.

1350 3.4.1 Element <AuthnRequest>

1351 To request that an identity provider issue an assertion with an authentication statement, a presenter
1352 authenticates to that identity provider (or relies on an existing security context) and sends it an

1352 <AuthnRequest> message that describes the properties that the resulting assertion needs to have to
1353 satisfy its purpose. Among these properties may be information that relates to the content of the
1354 assertion and/or information that relates to how the resulting <Response> message should be delivered
1355 to the requester. The process of authentication of the presenter may take place before, during, or after
1356 the initial delivery of the <AuthnRequest> message.

1353 The requester might not be the same as the presenter of the request if, for example, the requester is a
1354 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject
1355 so that the relying party can decide whether to provide a service.

1354 The <AuthnRequest> message SHOULD be signed or otherwise authenticated and integrity protected
1355 by the protocol binding used to deliver the message.

1355 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
1356 adds the following elements and attributes, all of which are optional in general, but may be required by
1357 specific profiles:

1356 <saml:Subject> [Optional]

1357 Specifies the requested subject of the resulting assertion(s). This may include one or more
1358 <saml:SubjectConfirmation> elements to indicate how and/or by whom the resulting assertions
1359 can be confirmed. For more information on this element, see Section 2.4.

1358 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
1359 requested subject. If no <saml:SubjectConfirmation> elements are included, then the
1360 presenter is presumed to be the only attesting entity required and the method is implied by the profile
1361 of use and/or the policies of the identity provider.

1359 <NameIDPolicy> [Optional]

1360 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
1361 then any type of identifier supported by the identity provider for the requested subject can be used,
1362 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

1361 <saml:Conditions> [Optional]

1362 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting
1363 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
1364 information in this element is used as input to the process of constructing the assertion, rather than
1365 as conditions on the use of the request itself. (For more information on this element, see Section
1366 2.5.)

1363 <RequestedAuthnContext> [Optional]

1364 Specifies the requirements, if any, that the requester places on the authentication context that
1365 applies to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for
1366 processing rules regarding this element.

1365 <Scoping> [Optional]

1366 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as
1367 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
1368 providers by the responder.

1367 ForceAuthn [Optional]

1368 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
1369 rely on a previous security context. If a value is not provided, the default is "false". However, if both
1370 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
1371 presenter unless the constraints of IsPassive can be met.

1369 `IsPassive` [Optional]

1370 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take
1371 control of the user interface from the requester and interact with the presenter in a noticeable
1372 fashion. If a value is not provided, the default is "false".

1371 `AssertionConsumerServiceIndex` [Optional]

1372 Indirectly identifies the location to which the `<Response>` message should be returned to the
1373 requester. It applies only to profiles in which the requester is different from the presenter, such as the
1374 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map
1375 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one
1376 possible mechanism. If omitted, then the identity provider MUST return the `<Response>` message
1377 to the default location associated with the requester for the profile of use. If the index specified is
1378 invalid, then the identity provider MAY return an error `<Response>` or it MAY use the default
1379 location. This attribute is mutually exclusive with the `AssertionConsumerServiceURL` and
1380 `ProtocolBinding` attributes.

1373 `AssertionConsumerServiceURL` [Optional]

1374 Specifies by value the location to which the `<Response>` message MUST be returned to the
1375 requester. The responder MUST ensure by some means that the value specified is in fact associated
1376 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing
1377 `<AuthnRequest>` message is another. This attribute is mutually exclusive with the
1378 `AssertionConsumerServiceIndex` attribute and is typically accompanied by the
1379 `ProtocolBinding` attribute.

1375 `ProtocolBinding` [Optional]

1376 A URI reference that identifies a SAML protocol binding to be used when returning the `<Response>`
1377 message. See [SAMLBind] for more information about protocol bindings and URI references defined
1378 for them. This attribute is mutually exclusive with the `AssertionConsumerServiceIndex`
1379 attribute and is typically accompanied by the `AssertionConsumerServiceURL` attribute.

1377 `AttributeConsumingServiceIndex` [Optional]

1378 Indirectly identifies information associated with the requester describing the SAML attributes the
1379 requester desires or requires to be supplied by the identity provider in the `<Response>` message.
1380 The identity provider MUST have a trusted means to map the index value in the attribute to
1381 information associated with the requester. [SAMLMeta] provides one possible mechanism. The
1382 identity provider MAY use this information to populate one or more `<saml:AttributeStatement>`
1383 elements in the assertion(s) it returns.

1379 `ProviderName` [Optional]

1380 Specifies the human-readable name of the requester for use by the presenter's user agent or the
1381 identity provider.

1381 See Section 3.4.1.4 for general processing rules regarding this message.

1382 The following schema fragment defines the `<AuthnRequest>` element and its `AuthnRequestType`
1383 complex type:

```
1383 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
1384 <complexType name="AuthnRequestType">
1385   <complexContent>
1386     <extension base="samlp:RequestAbstractType">
1387       <sequence>
1388         <element ref="saml:Subject" minOccurs="0"/>
1389         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
1390         <element ref="saml:Conditions" minOccurs="0"/>
1391         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>

```

```

1392         <element ref="samlp:Scoping" minOccurs="0"/>
1393     </sequence>
1394     <attribute name="ForceAuthn" type="boolean" use="optional"/>
1395     <attribute name="IsPassive" type="boolean" use="optional"/>
1396     <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
1397     <attribute name="AssertionConsumerServiceIndex"
1398 type="unsignedShort" use="optional"/>
1398     <attribute name="AssertionConsumerServiceURL" type="anyURI"
1399 use="optional"/>
1399     <attribute name="AttributeConsumingServiceIndex"
1400 type="unsignedShort" use="optional"/>
1400     <attribute name="ProviderName" type="string" use="optional"/>
1401 </extension>
1402 </complexContent>
1403 </complexType>

```

1404 3.4.1.1 Element <NameIDPolicy>

1405 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
1406 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

1406 Format [Optional]

1407 Specifies the URI reference corresponding to a name identifier format defined in this or another
1408 specification (see Section 8.3 for examples). The additional value of
1409 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use
1410 within this attribute to indicate a request that the resulting identifier be encrypted.

1408 SPNameQualifier [Optional]

1409 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace
1410 of a service provider other than the requester, or in the namespace of an affiliation group of service
1411 providers. See for example the definition of urn:oasis:names:tc:SAML:2.0:nameid-
1412 format:persistent in Section 8.3.7.

1410 AllowCreate [Optional]

1411 A Boolean value used to indicate whether [\[PE14\]the requester grants to](#) the identity provider ~~is~~
1412 ~~allowed~~, in the course of fulfilling the request, [permission](#) to create a new identifier ~~to represent the~~
1413 ~~principal or to associate an existing identifier representing the principal with the relying party~~. Defaults
1414 to "false". ~~When "false", the requester constrains the identity provider to only issue an assertion to it~~
1415 ~~if an acceptable identifier for the principal has already been established. Note that this does not~~
1416 ~~prevent the identity provider from creating such identifiers outside the context of this specific request~~
1417 ~~(for example, in advance for a large number of principals); if not present or the entire element is~~
1418 ~~omitted.~~

1412 [The AllowCreate attribute may be used by some deployments to influence the creation of state](#)
1413 [maintained by the identity provider pertaining to the use of a name identifier \(or any other persistent,](#)
1414 [uniquely identifying attributes\) by a particular relying party, for purposes such as dynamic identifier or](#)
1415 [attribute creation, tracking of consent, subsequent use of the Name Identifier Management protocol](#)
1416 [\(see Section 3.6\), or other related purposes.](#)

1413 [When "false", the requester tries to constrain the identity provider to issue an assertion only if such](#)
1414 [state has already been established or is not deemed applicable by the identity provider to the use of](#)
1415 [an identifier. Thus, this does not prevent the identity provider from assuming such information exists](#)
1416 [outside the context of this specific request \(for example, establishing it in advance for a large](#)
1417 [number of principals\).](#)

1414 [A value of "true" permits the identity provider to take any related actions it wishes to fulfill the](#)
1415 [request, subject to any other constraints imposed by the request and policy \(the IsPassive](#)
1416 [attribute, for example\).](#)

1415 [Generally, requesters cannot assume specific behavior from identity providers regarding the initial](#)
1416 [creation or association of identifiers on their behalf, as these are details left to implementations or](#)
1417 [deployments. Absent specific profiles governing the use of this attribute, it might be used as a hint to](#)
1418 [identity providers about the requester's intention to store the identifier or link it to a local value.](#)

1416 [A value of "false" might be used to indicate that the requester is not prepared or able to do so and](#)
1417 [save the identity provider wasted effort.](#)

1417 [Requesters that do not make specific use of this attribute SHOULD generally set it to "true" to](#)
1418 [maximize interoperability.](#)

1418 [The use of the AllowCreate attribute MUST NOT be used and SHOULD be ignored in conjunction](#)
1419 [with requests for or assertions issued with name identifiers with a `Format` of](#)
1420 [`urn:oasis:names:tc:SAML:2.0:nameid-format:transient` \(they preclude any such state](#)
1421 [in and of themselves\).](#)

1419 When this element is used, if the content is not understood by or acceptable to the identity provider, then
1420 a `<Response>` message element MUST be returned with an error `<Status>`, and MAY contain a
1421 second-level `<StatusCode>` of
1422 `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

1420 If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-`
1421 `format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any
1422 additional constraints due to the content of this element or the policies of the identity provider or
1423 principal.

1421 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`
1422 indicates that the resulting assertion(s) MUST contain `<EncryptedID>` elements instead of plaintext.
1423 The underlying name identifier's unencrypted form can be of any type supported by the identity provider
1424 for the requested subject. [\[PE6\]It is not possible for the service provider to specifically request that a](#)
1425 [particular kind of identifier be returned if it asks for encryption. The `<md:NameIDFormat>` metadata](#)
1426 [element \(see \[SAMLMeta\]\) or other out-of-band means MAY be used to determine what kind of identifier](#)
1427 [to encrypt and return.](#)

1422 [\[PE15\]When a `Format` defined in Section 8.3 other than `urn:oasis:names:TC:SAML:2.0:nameid-`](#)
1423 [`format:unspecified` or `urn:oasis:names:TC:SAML:2.0:nameid-format:encrypted` is used,](#)
1424 [then if the identity provider returns any assertions:](#)

- 1423 • [the `Format` value of the `<NameID>` within the `<Subject>` of any `<Assertion>` MUST be](#)
1424 [identical to the `Format` value supplied in the `<NameIDPolicy>`, and](#)
- 1424 • [if `SPNameQualifier` is not omitted in `<NameIDPolicy>`, the `SPNameQualifier` value of the](#)
1425 [`<NameID>` within the `<Subject>` of any `<Assertion>` MUST be identical to the](#)
1426 [`SPNameQualifier` value supplied in the `<NameIDPolicy>`.](#)

1425 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an
1426 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider
1427 (possibly specific to the service provider) require that an encrypted identifier be used.

1426 [\[PE14\]Note that if the requester wishes to permit the identity provider to establish a new identifier for the](#)
1427 [principal if none exists, it MUST include this element with the `AllowCreate` attribute set to "true".](#)
1428 [Otherwise, only a principal for whom the identity provider has previously established an identifier usable](#)
1429 [by the requester can be authenticated successfully. This is primarily useful in conjunction with the](#)
1430 [`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` `Format` value \(see Section-](#)
1431 [8.3.7\).](#)

1427 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
1428 complex type:

```

1428 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
1429 <complexType name="NameIDPolicyType">
1430   <attribute name="Format" type="anyURI" use="optional"/>
1431   <attribute name="SPNameQualifier" type="string" use="optional"/>
1432   <attribute name="AllowCreate" type="boolean" use="optional"/>
1433 </complexType>

```

1434 3.4.1.2 Element <Scoping>

1435 The <Scoping> element specifies the identity providers trusted by the requester to authenticate the
 1436 presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to
 1437 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
 1438 elements and attribute:

1436 ProxyCount [Optional]

1437 Specifies the number of proxying indirections permissible between the identity provider that receives
 1438 this <AuthnRequest> and the identity provider who ultimately authenticates the principal. A count
 1439 of zero permits no proxying, while omitting this attribute expresses no such restriction.

1438 <IDPList> [Optional]

1439 An advisory list of identity providers and associated information that the requester deems acceptable
 1440 to respond to the request.

1440 <RequesterID> [Zero or More]

1441 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate
 1442 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for
 1443 a description of entity identifiers.

1442 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
 1443 <Response> message with an error <Status> and a second-level <StatusCode> of
 1444 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or
 1445 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support
 1446 any of the specified identity providers.

1443 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```

1444 <element name="Scoping" type="samlp:ScopingType"/>
1445 <complexType name="ScopingType">
1446   <sequence>
1447     <element ref="samlp:IDPList" minOccurs="0"/>
1448     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
1449   </sequence>
1450   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
1451 </complexType>
1452 <element name="RequesterID" type="anyURI"/>

```

1453 3.4.1.3 Element <IDPList>

1454 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the
 1455 presenter. Its **IDPListType** complex type defines the following elements:

1455 <IDPEntry> [One or More]

1456 Information about a single identity provider.

1457 <GetComplete> [Optional]

1458 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to
 1459 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML

1459 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>
1460 element.

1460 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
1461 <element name="IDPList" type="samlp:IDPListType"/>  
1462 <complexType name="IDPListType">  
1463   <sequence>  
1464     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>  
1465     <element ref="samlp:GetComplete" minOccurs="0"/>  
1466   </sequence>  
1467 </complexType>  
1468 <element name="GetComplete" type="anyURI"/>
```

1469 **3.4.1.3.1 Element <IDPEntry>**

1470 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the
1471 presenter. Its **IDPEntryType** complex type defines the following attributes:

1471 ProviderID [Required]

1472 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

1473 Name [Optional]

1474 A human-readable name for the identity provider.

1475 Loc [Optional]

1476 A URI reference representing the location of a profile-specific endpoint supporting the authentication
1477 request protocol. The binding to be used must be understood from the profile of use.

1477 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
1478 <element name="IDPEntry" type="samlp:IDPEntryType"/>  
1479 <complexType name="IDPEntryType">  
1480   <attribute name="ProviderID" type="anyURI" use="required"/>  
1481   <attribute name="Name" type="string" use="optional"/>  
1482   <attribute name="Loc" type="anyURI" use="optional"/>  
1483 </complexType>
```

1484 **3.4.1.4 Processing Rules**

1485 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
1486 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
1487 kinds of input and output are required or prohibited. The following processing rules apply as invariant
1488 behavior across any profile of this protocol exchange. All other processing rules associated with the
1489 underlying request and response messages **MUST** also be observed.

1486 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing
1487 one or more assertions that meet the specifications defined by the request, or with a <Response>
1488 message containing a <Status> describing the error that occurred. The responder **MAY** conduct
1489 additional message exchanges with the presenter as needed to initiate or complete the authentication
1490 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
1491 the next section, this includes proxying the request by directing the presenter to another identity provider
1492 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to
1493 authenticate the presenter to the original responder, in effect using SAML as the authentication
1494 mechanism.

1487 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if
1488 prevented from providing an assertion by policies in effect at the identity provider (for example the

1488 intended subject has prohibited the identity provider from providing assertions to the relying party), then
1489 it MUST return a <Response> with an error <Status>, and MAY return a second-level <StatusCode>
1490 of urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
1491 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

1489 If the <saml:Subject> element in the request is present, then the resulting assertions'
1490 <saml:Subject> MUST **strongly match** the request <saml:Subject>, as described in Section 3.3.4,
1491 except that the identifier MAY be in a different format if specified by <NameIDPolicy>. In such a case,
1492 the identifier's physical content MAY be different, but it MUST refer to the same principal.

1490 All of the content defined specifically within <AuthnRequest> is optional, although some may be
1491 required by certain profiles. In the absence of any specific content at all, the following behavior is
1492 implied:

- 1491 • The assertion(s) returned MUST contain a <saml:Subject> element that represents the
1492 presenter. The identifier type and format are determined by the identity provider. At least one
1493 statement in at least one assertion MUST be a <saml:AuthnStatement> that describes the
1494 authentication performed by the responder or authentication service associated with it.
- 1492 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the
1493 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation
1494 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- 1493 • The resulting assertion(s) MUST contain a <saml:AudienceRestriction> element
1494 referencing the requester as an acceptable relying party. Other audiences MAY be included as
1495 deemed appropriate by the identity provider.

1494 3.4.1.5 Proxying

1495 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or
1496 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
1497 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
1498 <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in
1499 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
1500 identity provider.

1496 Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the
1497 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
1498 proxying provider can issue an assertion of its own in response to the original <AuthnRequest>,
1499 completing the overall message exchange. Both the proxying and authenticating identity providers MAY
1500 include constraints on proxying activity in the messages and assertions they issue, as described in
1501 previous sections and below.

1497 The requester can influence proxy behavior by including a <Scoping> element where the provider sets
1498 a desired ProxyCount value and/or indicates a list of preferred identity providers which may be proxied
1499 by including an ordered <IDPList> of preferred providers.

1498 An identity provider can control secondary use of its assertions by proxying identity providers using a
1499 <ProxyRestriction> element in the assertions it issues.

1499 3.4.1.5.1 Proxying Processing Rules

1500 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is
1501 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
1502 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

1501 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity
 1502 provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of
 1503 `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly
 1504 authenticate the presenter.

1502 If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the
 1503 proxying identity provider MUST include equivalent or stricter forms of all the information included in the
 1504 original request (such as authentication context policy). Note, however, that the proxying provider is free
 1505 to specify whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

1503 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
 1504 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for
 1505 example) will be honored by the authenticating provider.

1504 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less
 1505 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new
 1506 request SHOULD contain a `<ProxyCount>` attribute.

1505 If an `<IDPList>` was specified in the original request, the new request MUST also contain an
 1506 `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the
 1507 `<IDPList>`, but MUST NOT remove any from the list.

1506 The authentication request and response are processed in normal fashion, in accordance with the rules
 1507 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
 1508 provider (in the case of SAML by delivering a `<Response>`), the following steps are followed:

- 1507 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
 1508 relevant information from the original assertion or non-SAML equivalent.
- 1508 • The new assertion's `<saml:Subject>` MUST contain an identifier that satisfies the original
 1509 requester 's preferences, as defined by its `<NameIDPolicy>` element.
- 1509 • The `<saml:AuthnStatement>` in the new assertion MUST include a `<saml:AuthnContext>`
 1510 element containing a `<saml:AuthenticatingAuthority>` element referencing the identity
 1511 provider to which the proxying identity provider referred the presenter. If the original assertion
 1512 contains `<saml:AuthnContext>` information that includes one or more
 1513 `<saml:AuthenticatingAuthority>` elements, those elements SHOULD be included in the
 1514 new assertion, with the new element placed after them.
- 1510 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
 1511 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
 1512 consistent over time across different requests. The value MUST not conflict with values used or
 1513 generated by other SAML providers.
- 1511 • Any other `<saml:AuthnContext>` information MAY be copied, translated, or omitted in
 1512 accordance with the policies of the proxying identity provider, provided that the original
 1513 requirements dictated by the requester are met.

1512 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,
 1513 and this request is equally or less strict than the original request (as determined by the proxying identity
 1514 provider), the identity provider MAY skip the creation of a new `<AuthnRequest>` to the authenticating
 1515 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML
 1516 equivalent it received is still valid).

1513 3.5 Artifact Resolution Protocol

1514 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
1515 transported in a SAML binding by reference instead of by value. Both requests and responses can be
1516 obtained by reference using this specialized protocol. A message sender, instead of binding a message
1517 to a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can
1518 take a variety of forms, but must support a means by which the receiver can determine who sent it. If the
1519 receiver wishes, it can then use this protocol in conjunction with a different (generally synchronous)
1520 SAML binding protocol to resolve the artifact into the original protocol message.

1515 The most common use for this mechanism is with bindings that cannot easily carry a message because
1516 of size constraints, or to enable a message to be communicated via a secure channel between the SAML
1517 requester and responder, avoiding the need for a signature.

1516 Depending on the characteristics of the underlying message being passed by reference, the artifact
1517 resolution protocol MAY require protections such as mutual authentication, integrity protection,
1518 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
1519 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
1520 by any party.

1517 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
1518 as if the message so obtained had been sent originally in place of the artifact.

1518 3.5.1 Element <ArtifactResolve>

1519 The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an
1520 <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message.
1521 The original transmission of the artifact is governed by the specific protocol binding that is being used;
1522 see [SAMLBind] for more information on the use of artifacts in bindings.

1520 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
1521 protected by the protocol binding used to deliver the message.

1521 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
1522 adds the following element:

1522 <Artifact> [Required]

1523 The artifact value that the requester received and now wishes to translate into the protocol message
1524 it represents. See [SAMLBind] for specific artifact format information.

1524 The following schema fragment defines the <ArtifactResolve> element and its
1525 **ArtifactResolveType** complex type:

```
1525 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
1526 <complexType name="ArtifactResolveType">
1527   <complexContent>
1528     <extension base="samlp:RequestAbstractType">
1529       <sequence>
1530         <element ref="samlp:Artifact"/>
1531       </sequence>
1532     </extension>
1533   </complexContent>
1534 </complexType>
1535 <element name="Artifact" type="string"/>
```

1536 3.5.2 Element <ArtifactResponse>

1537 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
1538 message element. This element is of complex type **ArtifactResponseType**, which extends
1539 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
1540 message being returned. This wrapped message element can be a request or a response.

1538 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
1539 protected by the protocol binding used to deliver the message.

1539 The following schema fragment defines the <ArtifactResponse> element and its
1540 **ArtifactResponseType** complex type:

```
1540 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>  
1541 <complexType name="ArtifactResponseType">  
1542   <complexContent>  
1543     <extension base="samlp:StatusResponseType">  
1544       <sequence>  
1545         <any namespace="##any" processContents="lax" minOccurs="0"/>  
1546       </sequence>  
1547     </extension>  
1548   </complexContent>  
1549 </complexType>
```

1550 3.5.3 Processing Rules

1551 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
1552 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>
1553 element with no embedded message. In both cases, the <Status> element MUST include a
1554 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A
1555 response message with no embedded message inside it is termed an empty response in the remainder
1556 of this section.

1552 The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent
1553 request with the same artifact by any requester results in an empty response as described above.

1553 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles,
1554 MAY be intended for consumption by any party that receives it and can respond appropriately. In most
1555 other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued
1556 MUST be associated with the intended recipient of the message that the artifact represents. If the artifact
1557 issuer receives an <ArtifactResolve> message from a requester that cannot authenticate itself as
1558 the original intended recipient, then the artifact issuer MUST return an empty response.

1554 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
1555 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
1556 return it in an <ArtifactResolve> message to the issuer.

1555 Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of
1556 the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message
1557 will contain its own message identifier, and in the case of an embedded response, may contain a
1558 different InResponseTo value that corresponds to the original request message to which the embedded
1559 message is responding.

1556 All other processing rules associated with the underlying request and response messages MUST be
1557 observed.

1557 3.6 Name Identifier Management Protocol

1558 After establishing a name identifier for a principal, an identity provider wishing to change the value
1559 [\[PE12\]and/or format](#) of the identifier that it will use when referring to the principal, or to indicate that a
1560 name identifier will no longer be used to refer to the principal, informs service providers of the change by
1561 sending them a <ManageNameIDRequest> message.

1559 A service provider also uses this message to register or change the `SPProvidedID` value to be included
1560 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
1561 identifier between itself and the identity provider.

1560 ~~[\[PE14\]Note that this protocol is typically not used with "transient" name identifiers, since their value is not](#)~~
1561 ~~[intended to be managed on a long term basis.This protocol MUST NOT be used in conjunction with the](#)~~
1562 ~~[urn:oasis:names:tc:SAML:2.0:nameidformat:transient <NameID> Format.](#)~~

1561 3.6.1 Element <ManageNameIDRequest>

1562 A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name
1563 identifier or to indicate the termination of the use of a name identifier.

1563 The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity
1564 protected by the protocol binding used to deliver the message.

1564 This message has the complex type **ManageNameIDRequestType**, which extends
1565 **RequestAbstractType** and adds the following elements:

1565 <saml:NameID> or <saml:EncryptedID> [Required]

1566 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
1567 principal as currently recognized by the identity and service providers prior to this request. (For
1568 more information on these elements, see Section 2.2.)

1567 <NewID> or <NewEncryptedID> or <Terminate> [Required]

1568 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
1569 requesting provider concerning this principal, or an indication that the use of the old identifier has
1570 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
1571 appear in subsequent <NameID> elements in the `SPProvidedID` attribute. If the requester is the
1572 identity provider, the new value will appear in subsequent <NameID> elements as the element's
1573 content. [\[PE12\]In either case, if the <NewEncryptedID> is used, its encrypted content is just a](#)
1574 [<NewID> element containing only the new value for the identifier \(format and qualifiers cannot be](#)
1575 [changed once established\).](#)

1569 The following schema fragment defines the <ManageNameIDRequest> element and its
1570 **ManageNameIDRequestType** complex type:

```
1570 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
1571 <complexType name="ManageNameIDRequestType">
1572   <complexContent>
1573     <extension base="samlp:RequestAbstractType">
1574       <sequence>
1575         <choice>
1576           <element ref="saml:NameID"/>
1577           <element ref="saml:EncryptedID"/>
1578         </choice>
1579         <choice>
1580           <element ref="samlp:NewID"/>
1581           <element ref="samlp:NewEncryptedID"/>
1582           <element ref="samlp:Terminate"/>
1583         </choice>

```

```

1584         </sequence>
1585     </extension>
1586 </complexContent>
1587 </complexType>
1588 <element name="NewID" type="string"/>
1589 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
1590 <element name="Terminate" type="samlp:TerminateType"/>
1591 <complexType name="TerminateType"/>

```

1592 3.6.2 Element <ManageNameIDResponse>

1593 The recipient of a <ManageNameIDRequest> message MUST respond with a
 1594 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional
 1595 content.

1594 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity
 1595 protected by the protocol binding used to deliver the message.

1595 The following schema fragment defines the <ManageNameIDResponse> element:

```

1596 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>

```

1597 3.6.3 Processing Rules

1598 If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize,
 1599 the responding provider MUST respond with an error <Status> and MAY respond with a second-level
 1600 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

1599 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the
 1600 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of
 1601 an identity provider) it will no longer issue assertions to the service provider about the principal. The
 1602 receiving provider can perform any maintenance with the knowledge that the relationship represented by
 1603 the name identifier has been terminated. [\[PE8\] In general it SHOULD NOT invalidate any active
 1604 session\(s\) of the principal for whom the relationship has been terminated. If the receiving provider is an
 1605 identity provider, it SHOULD NOT invalidate any active session\(s\) of the principal established with other
 1606 service providers. A requesting provider MAY send a <LogoutRequest> message prior to initiating a
 1607 name identifier termination by sending a <ManageNameIDRequest> message if that is the requesting
 1608 provider's intent \(e.g., the name identifier termination is initiated via an administrator who wished to
 1609 terminate all user activity\). The requesting provider MUST NOT send a <LogoutRequest> message
 1610 after the <ManageNameIDRequest> message is sent. It can choose to invalidate the active session\(s\) of
 1611 a principal for whom a relationship has been terminated.](#)

1600 [\[PE14\] If the receiving provider is maintaining state associated with the name identifier, such as the
 1601 value of the identifier itself \(in the case of a pair-wise identifier\), an SPProvidedID value, the sender's
 1602 consent to the identifier's creation/use, etc., then the receiver can perform any maintenance with the
 1603 knowledge that the relationship represented by the name identifier has been terminated.](#)

1601 [Any subsequent operations performed by the receiver on behalf of the sender regarding the principal \(for
 1602 example, a subsequent <AuthnRequest>\) SHOULD be carried out in a manner consistent with the
 1603 absence of any previous state.](#)

1602 [Termination is potentially the cleanup step for any state management behavior triggered by the use of
 1603 the AllowCreate attribute in the Authentication Request protocol \(see Section 3.4\). Deployments that
 1604 do not make use of that attribute are likely to avoid the use of the <Terminate> element or would treat
 1605 it as a purely advisory matter.](#)

1603 [Note that in most cases \(a notable exception being the rules surrounding the `SPProvidedID` attribute\),](#)
1604 [there are no requirements on either identity providers or service providers regarding the creation or use](#)
1605 [of persistent state. Therefore, no explicit behavior is mandated when the `<Terminate>` element is](#)
1606 [received. However, if persistent state is present pertaining to the use of an identifier \(such as if an](#)
1607 [`SPProvidedID` attribute was attached\), the `<Terminate>` element provides a clear indication that this](#)
1608 [state SHOULD be deleted \(or marked as obsolete in some fashion\).](#)

1604 If the service provider requests that its identifier for the principal be changed by including a `<NewID>` (or
1605 `<NewEncryptedID>`) element, the identity provider MUST include the element's content as the
1606 `SPProvidedID` when subsequently communicating to the service provider regarding this principal.

1605 If the identity provider requests that its identifier for the principal be changed by including a `<NewID>` (or
1606 `<NewEncryptedID>`) element, the service provider MUST use the element's content as the
1607 `<saml:NameID>` element content when subsequently communicating with the identity provider
1608 regarding this principal.

1606 Note that neither, either, or both of the original and new identifier MAY be encrypted (using the
1607 `<EncryptedID>` and `<NewEncryptedID>` elements).

1607 In any case, the `<saml:NameID>` content in the request and its associated `SPProvidedID` attribute
1608 MUST contain the most recent name identifier information established between the providers for the
1609 principal.

1608 In the case of an identifier with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`
1609 `format:persistent`, the `NameQualifier` attribute MUST contain the unique identifier of the identity
1610 provider that created the identifier. If the identifier was established between the identity provider and an
1611 affiliation group of which the service provider is a member, then the `SPNameQualifier` attribute MUST
1612 contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of
1613 the service provider. These attributes MAY be omitted if they would otherwise match the value of the
1614 containing protocol message's `<Issuer>` element, but this is NOT RECOMMENDED due to the
1615 opportunity for confusion.

1609 Changes to these identifiers may take a potentially significant amount of time to propagate through the
1610 systems at both the requester and the responder. Implementations might wish to allow each party to
1611 accept either identifier for some period of time following the successful completion of a name identifier
1612 change. Not doing so could result in the inability of the principal to access resources.

1610 All other processing rules associated with the underlying request and response messages MUST be
1611 observed.

1611 **3.7 Single Logout Protocol**

1612 The single logout protocol provides a message exchange protocol by which all sessions provided by a
1613 particular session authority are near-simultaneously terminated. The single logout protocol is used either
1614 when a principal logs out at a session participant or when the principal logs out directly at the
1615 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
1616 the logout event can be indicated through the `Reason` attribute.

1617
1618 The principal may have established authenticated sessions with both the session authority and individual
1619 session participants, based on assertions containing authentication statements supplied by the session
1620 authority.

1621
1622 When the principal invokes the single logout process at a session participant, the session participant
1623 MUST send a `<LogoutRequest>` message to the session authority that provided the assertion
1624 containing the authentication statement related to that session at the session participant.

1625

1613 When either the principal invokes a logout at the session authority, or a session participant sends a
1614 logout request to the session authority specifying that principal, the session authority SHOULD send a
1615 <LogoutRequest> message to each session participant to which it provided assertions containing
1616 authentication statements under its current session with the principal, with the exception of the session
1617 participant that sent the <LogoutRequest> message to the session authority. It SHOULD attempt to
1618 contact as many of these participants as it can using this protocol, terminate its own session with the
1619 principal, and finally return a <LogoutResponse> message to the requesting session participant, if any.

1614 3.7.1 Element <LogoutRequest>

1615 A session participant or session authority sends a <LogoutRequest> message to indicate that a
1616 session has been terminated.

1616 The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected
1617 by the protocol binding used to deliver the message.

1617 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and
1618 adds the following elements and attributes:

1618 NotOnOrAfter [Optional]

1619 The time at which the request expires, after which the recipient may discard the message. The time
1620 value is encoded in UTC, as described in Section 1.3.3.

1620 Reason [Optional]

1621 An indication of the reason for the logout, in the form of a URI reference. [\[PE10\] The Reason](#)
1622 [attribute is specified as a string in the schema. This specification further restricts the schema by](#)
1623 [requiring that the Reason attribute MUST be in the form of a URI reference.](#)

1622 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

1623 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as
1624 currently recognized by the identity and service providers prior to this request. (For more information
1625 on this element, see Section 2.2.)

1624 <SessionIndex> [Optional]

1625 [\[PE38\] The index of the session between the principal identified by the <saml:BaseID>,](#)
1626 [<saml:NameID>, or <saml:EncryptedID> element, and the session authority. This must](#)
1627 [correlate to the SessionIndex attribute, if any, in the <saml:AuthnStatement> of the assertion](#)
1628 [used to establish the session that is being terminated. The identifier that indexes this session at the](#)
1629 [message recipient.](#)

1626 The following schema fragment defines the <LogoutRequest> element and associated
1627 **LogoutRequestType** complex type:

```
1627 <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
1628   <complexType name="LogoutRequestType">
1629     <complexContent>
1630       <extension base="samlp:RequestAbstractType">
1631         <sequence>
1632           <choice>
1633             <element ref="saml:BaseID"/>
1634             <element ref="saml:NameID"/>
1635             <element ref="saml:EncryptedID"/>
1636           </choice>
1637           <element ref="samlp:SessionIndex" minOccurs="0"
1638 maxOccurs="unbounded"/>
1638         </sequence>
1639         <attribute name="Reason" type="string" use="optional"/>

```

```
1640         <attribute name="NotOnOrAfter" type="dateTime"
1641 use="optional"/>
1642     </extension>
1643 </complexContent>
1644 </complexType>
1645 <element name="SessionIndex" type="string"/>
```

1645 3.7.2 Element <LogoutResponse>

1646 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message,
1647 of type **StatusResponseType**, with no additional content specified.

1647 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
1648 protected by the protocol binding used to deliver the message.

1648 The following schema fragment defines the <LogoutResponse> element:

```
1649 <element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

1650 3.7.3 Processing Rules

1651 The message sender MAY use the Reason attribute to indicate the reason for sending the
1652 <LogoutRequest>. The following values are defined by this specification for use by all message
1653 senders; other values MAY be agreed on between participants:

1652 urn:oasis:names:tc:SAML:2.0:logout:user

1653 Specifies that the message is being sent because the principal wishes to terminate the indicated
1654 session.

1654 urn:oasis:names:tc:SAML:2.0:logout:admin

1655 Specifies that the message is being sent because an administrator wishes to terminate the indicated
1656 session for that principal.

1656 All other processing rules associated with the underlying request and response messages MUST be
1657 observed.

1657 Additional processing rules are provided in the following sections.

1658 3.7.3.1 Session Participant Rules

1659 When a session participant receives a <LogoutRequest> message, the session participant MUST
1660 authenticate the message. If the sender is the authority that provided an assertion containing an
1661 authentication statement linked to the principal's current session, the session participant MUST invalidate
1662 the principal's session(s) referred to by the <saml:BaseID>, <saml:NameID>, or
1663 <saml:EncryptedID> element, and any <SessionIndex> elements supplied in the message. If no
1664 <SessionIndex> elements are supplied, then all sessions associated with the principal MUST be
1665 invalidated.

1666 The session participant MUST apply the logout request message to any assertion that meets the
1667 following conditions, even if the assertion arrives after the logout request:

- 1660 • The subject of the assertion **strongly matches** the <saml:BaseID>, <saml:NameID>, or
1661 <saml:EncryptedID> element in the <LogoutRequest>, as defined in Section 3.3.4.
- 1661 • The SessionIndex attribute of one of the assertion's authentication statements matches one of
1662 the <SessionIndex> elements specified in the logout request, or the logout request contains no
1663 <SessionIndex> elements.

- 1662 • The assertion would otherwise be valid, based on the time conditions specified in the assertion
1663 itself (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject
1664 confirmation data).
- 1663 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on
1664 the message).

1664 **Note:** This rule is intended to prevent a situation in which a session participant receives
1665 a logout request targeted at a single, or multiple, assertion(s) (as identified by the
1666 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid –
1667 assertion(s) targeted by the logout request. It should honor the logout request until the
1668 logout request itself may be discarded (the `NotOnOrAfter` value on the request has
1669 been exceeded) or the assertion targeted by the logout request has been received and
1670 has been handled appropriately.

1665 3.7.3.2 Session Authority Rules

1666 When a session authority receives a `<LogoutRequest>` message, the session authority **MUST**
1667 authenticate the sender. If the sender is a session participant to which the session authority provided an
1668 assertion containing an authentication statement for the current session, then the session authority
1669 **SHOULD** do the following in the specified order:

- 1667 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session
1668 authority proxied the principal's authentication, unless the second authority is the originator of the
1669 `<LogoutRequest>`.
- 1668 • Send a `<LogoutRequest>` message to each session participant for which the session authority
1669 provided assertions in the current session, *other than* the originator of a current
1670 `<LogoutRequest>`.
- 1669 • Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`,
1670 or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout
1671 request message.

1670 If the session authority successfully terminates the principal's session with respect to itself, then it **MUST**
1671 respond to the original requester, if any, with a `<LogoutResponse>` message containing a top-level
1672 status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it **MUST**
1673 respond with a `<LogoutResponse>` message containing a top-level status code indicating the error.
1674 Thus, the top-level status indicates the state of the logout operation only with respect to the session
1675 authority itself.

1671 The session authority **SHOULD** attempt to contact each session participant using any applicable/usable
1672 protocol binding, even if one or more of these attempts fails or cannot be attempted (for example
1673 because the original request takes place using a protocol binding that does not enable the logout to be
1674 propagated to all participants).

1672 In the event that not all session participants successfully respond to these `<LogoutRequest>`
1673 messages (or if not all participants can be contacted), then the session authority **MUST** include in its
1674 `<LogoutResponse>` message a second-level status code of
1675 `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session
1676 participants successfully responded with confirmation of the logout.

1673 Note that a session authority **MAY** initiate a logout for reasons other than having received a
1674 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 1674 • If some timeout period was agreed out-of-band with an individual session participant, the session
1675 authority **MAY** send a `<LogoutRequest>` to that individual participant alone.

- 1675 • An agreed global timeout period has been exceeded.
- 1676 • The principal or some other trusted entity has requested logout of the principal directly at the
1677 session authority.
- 1677 • The session authority has determined that the principal's credentials may have been compromised.

1678 When constructing a logout request message, the session authority MUST set the value of the
1679 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message,
1680 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time
1681 value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most
1682 recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the
1683 logout request).

1679 In addition to the values specified in Section [\[E01\] 3.7.33-6-3](#) for the `Reason` attribute, the following
1680 values are also available for use by the session authority only:

1680 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

1681 Specifies that the message is being sent because of the global session timeout interval period
1682 being exceeded.

1682 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

1683 Specifies that the message is being sent because a timeout interval period agreed between a
1684 participant and the session authority has been exceeded.

1684 3.8 Name Identifier Mapping Protocol

1685 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
1686 identifier for the same principal in a particular format or federation namespace, it can send a request to
1687 the identity provider using this protocol.

1686 For example, a service provider that wishes to communicate with another service provider with whom it
1687 does not share an identifier for the principal can use an identity provider that shares an identifier for the
1688 principal with both service providers to map from its own identifier to a new identifier, generally
1689 encrypted, with which it can communicate with the second service provider.

1687 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
1688 `<saml:EncryptedID>` element unless a specific deployment dictates such protection is unnecessary.

1688 3.8.1 Element `<NameIDMappingRequest>`

1689 To request an alternate name identifier for a principal from an identity provider, a requester sends an
1690 `<NameIDMappingRequest>` message. This message has the complex type
1691 **`NameIDMappingRequestType`**, which extends **`RequestAbstractType`** and adds the following elements:

1690 `<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

1691 The identifier and associated descriptive data that specify the principal as currently recognized by
1692 the requester and the responder. (For more information on this element, see Section 2.2.)

1692 `<NameIDPolicy>` [Required]

1693 The requirements regarding the format and optional name qualifier for the identifier to be returned.

1694 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
1695 binding used to deliver the message.

1695 The following schema fragment defines the <NameIDMappingRequest> element and its
1696 **NameIDMappingRequestType** complex type:

```
1696 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
1697 <complexType name="NameIDMappingRequestType">
1698   <complexContent>
1699     <extension base="samlp:RequestAbstractType">
1700       <sequence>
1701         <choice>
1702           <element ref="saml:BaseID"/>
1703           <element ref="saml:NameID"/>
1704           <element ref="saml:EncryptedID"/>
1705         </choice>
1706         <element ref="samlp:NameIDPolicy"/>
1707       </sequence>
1708     </extension>
1709   </complexContent>
1710 </complexType>
```

1711 3.8.2 Element <NameIDMappingResponse>

1712 The recipient of a <NameIDMappingRequest> message MUST respond with a
1713 <NameIDMappingResponse> message. This message has the complex type
1714 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following
1715 element:

1713 <saml:NameID> or <saml:EncryptedID> [Required]

1714 The identifier and associated attributes that specify the principal in the manner requested, usually in
1715 encrypted form. (For more information on this element, see Section 2.2.)

1715 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
1716 binding used to deliver the message.

1716 The following schema fragment defines the <NameIDMappingResponse> element and its
1717 **NameIDMappingResponseType** complex type:

```
1717 <element name="NameIDMappingResponse"
1718 type="samlp:NameIDMappingResponseType"/>
1718 <complexType name="NameIDMappingResponseType">
1719   <complexContent>
1720     <extension base="samlp:StatusResponseType">
1721       <choice>
1722         <element ref="saml:NameID"/>
1723         <element ref="saml:EncryptedID"/>
1724       </choice>
1725     </extension>
1726   </complexContent>
1727 </complexType>
```

1728 3.8.3 Processing Rules

1729 If the responder does not recognize the principal identified in the request, it MAY respond with an error
1730 <Status> containing a second-level <StatusCode> of
1731 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

1730 At the responder's discretion, the
1731 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to
1732 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

1731 All other processing rules associated with the underlying request and response messages MUST be
1732 observed.

1732 4 SAML Versioning

1733 The SAML specification set is versioned in two independent ways. Each is discussed in the following
1734 sections, along with processing rules for detecting and handling version differences. Also included are
1735 guidelines on when and why specific version information is expected to change in future revisions of the
1736 specification.

1734 When version information is expressed as both a Major and Minor version, it is expressed in the form
1735 *Major.Minor*. The version number *Major_B.Minor_B* is higher than the version number *Major_A.Minor_A* if and
1736 only if:

1735 $(Major_B > Major_A) \text{ OR } ((Major_B = Major_A) \text{ AND } (Minor_B > Minor_A))$

1736 4.1 SAML Specification Set Version

1737 Each release of the SAML specification set will contain a major and minor version designation describing
1738 its relationship to earlier and later versions of the specification set. The version will be expressed in the
1739 content and filenames of published materials, including the specification set documents and XML
1740 schema documents. There are no normative processing rules surrounding specification set versioning,
1741 since it merely encompasses the collective release of normative specification documents which
1742 themselves contain processing rules.

1738 The overall size and scope of changes to the specification set documents will informally dictate whether
1739 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards
1740 compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain
1741 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
1742 revision.

1739 4.1.1 Schema Version

1740 As a non-normative documentation mechanism, any XML schema documents published as part of the
1741 specification set will contain a `version` attribute on the `<xs:schema>` element whose value is in the
1742 form *Major.Minor*, reflecting the specification set version in which it has been published. Validating
1743 implementations MAY use the attribute as a means of distinguishing which version of a schema is being
1744 used to validate messages, or to support multiple versions of the same logical schema.

1741 4.1.2 SAML Assertion Version

1742 The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of
1743 the assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be
1744 construed so as to document the syntax, semantics, and processing rules of the assertions of the same
1745 version. That is, specification set version 1.0 describes assertion version 1.0, and so on.

1743 There is explicitly NO relationship between the assertion version and the target XML namespace
1744 specified for the schema definitions for that assertion version.

1744 The following processing rules apply:

- 1745 • A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion
1746 version number not supported by the authority.
- 1747 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
1748 supported by the relying party.

- 1749
- 1750
- 1751
- 1752
- 1753
- 1754
- 1755
- 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, all assertions that share a major assertion version number MUST share the same general processing rules and semantics, and MAY be treated in a uniform way by an implementation. For example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

1756 4.1.3 SAML Protocol Version

1757 The various SAML protocols' request and response elements contain an attribute for expressing the
1758 major and minor version of the request or response message using a string of the form *Major.Minor*.
1759 Each version of the SAML specification set will be construed so as to document the syntax, semantics,
1760 and processing rules of the protocol messages of the same version. That is, specification set version 1.0
1761 describes request and response version V1.0, and so on.

1758 There is explicitly NO relationship between the protocol version and the target XML namespace specified
1759 for the schema definitions for that protocol version.

1759 The version numbers used in SAML protocol request and response elements will match for any particular
1760 revision of the SAML specification set.

1760 4.1.3.1 Request Version

1761 The following processing rules apply to requests:

- 1762
- 1763
- A SAML requester SHOULD issue requests with the highest request version supported by both the SAML requester and the SAML responder.
 - If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD assume that the responder supports requests with the highest request version supported by the requester.
 - A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version number matching a response version number that the requester does not support.
 - A SAML responder MUST reject any request with a major request version number not supported by the responder.
 - A SAML responder MAY process or MAY reject any request whose minor request version number is higher than the highest supported request version that it supports. However, all requests that share a major request version number MUST share the same general processing rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)
- 1764
- 1765
- 1766
- 1767
- 1768
- 1769
- 1770
- 1771
- 1772
- 1773
- 1774
- 1775
- 1776
- 1777

1778 4.1.3.2 Response Version

1779 The following processing rules apply to responses:

- 1780
- 1781
- A SAML responder MUST NOT issue a response message with a response version number higher than the request version number of the corresponding request message.
 - A SAML responder MUST NOT issue a response message with a major response version number lower than the major request version number of the corresponding request message except to report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 1782
- 1783
- 1784

- 1785 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
1786 top-level <StatusCode> value of
1787 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch, and MAY result in reporting
1788 one of the following second-level values:
1789 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,
1790 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or
1791 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

1786 4.1.3.3 Permissible Version Combinations

1787 Assertions of a particular major version appear only in response messages of the same major version, as
1788 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
1789 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1
1790 response message, if the appropriate assertion schema is referenced during namespace importation. But
1791 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major
1792 versions.

1788 4.2 SAML Namespace Version

1789 XML schema documents published as part of the specification set contain one or more target
1790 namespaces into which the type, element, and attribute definitions are placed. Each namespace is
1791 distinct from the others, and represents, in shorthand, the structural and syntactic definitions that make
1792 up that part of the specification.

1790 The namespace URI references defined by the specification set will generally contain version information
1791 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST
1792 correspond to the major and minor version of the specification set in which the namespace is first
1793 introduced and defined. This information is not typically consumed by an XML processor, which treats
1794 the namespace opaquely, but is intended to communicate the relationship between the specification set
1795 and the namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers
1796 that are listed in Section 8.

1791 As a general rule, implementers can expect the namespaces and the associated schema definitions
1792 defined by a major revision of the specification set to remain valid and stable across minor revisions of
1793 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,
1794 but this is expected to be rare. In such cases, the older namespaces and their associated definitions
1795 should be expected to remain valid until a major specification set revision.

1792 4.2.1 Schema Evolution

1793 In general, maintaining namespace stability while adding or changing the content of a schema are
1794 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
1795 older implementations will react to any given change, making forward compatibility difficult to achieve.
1796 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of
1797 namespace stability. Except in special circumstances (for example, to correct major deficiencies or to fix
1798 errors), implementations should expect forward-compatible schema changes in minor revisions, allowing
1799 new messages to validate against older schemas.

1794 Implementations SHOULD expect and be prepared to deal with new extensions and message types in
1795 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types
1796 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such
1797 extensions gracefully when they are encountered in contexts that dictate mandatory semantics.
1798 Examples include new query, statement, or condition types.

1795 5 SAML and XML Signature Syntax and Processing

1796 SAML assertions and SAML protocol request and response messages may be signed, with the following
1797 benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the
1798 asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-
1799 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the
1800 message originator supports message integrity, authentication of message origin to a destination, and, if
1801 the signature is based on the originator's public-private key pair, non-repudiation of origin.

1797 A digital signature is not always required in SAML. For example, in some circumstances, signatures may
1798 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing
1799 protocol response message. "Inherited" signatures should be used with care when the contained object
1800 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context
1801 must be retained to allow validation, exposing the XML content and adding potentially unnecessary
1802 overhead. As another example, the SAML relying party or SAML requester may have obtained an
1803 assertion or protocol message from the SAML asserting party or SAML responder directly (with no
1804 intermediaries) through a secure channel, with the asserting party or SAML responder having
1805 authenticated to the relying party or SAML responder by some means other than a digital signature.

1798 Many different techniques are available for "direct" authentication and secure channel establishment
1799 between two parties. The list includes TLS/SSL (see [RFC 2246]/[SSL3]), HMAC, password-based
1800 mechanisms, and so on. In addition, the applicable security requirements depend on the communicating
1801 applications and the nature of the assertion or message transported. It is RECOMMENDED that, in all
1802 other contexts, digital signatures be used for assertions and request and response messages.
1803 Specifically:

- 1799 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting
1800 party SHOULD be signed by the SAML asserting party.
- 1800 • A SAML protocol message arriving at a destination from an entity other than the originating sender
1801 SHOULD be signed by the sender.
- 1801 • Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects
1802 that contain SAML documents. Caveats about retaining context and interoperability apply. XML
1803 Signatures are intended to be the primary SAML signature mechanism, but this specification
1804 attempts to ensure compatibility with profiles that may require other mechanisms.
- 1802 • Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST
1803 be enveloped.

1803 5.1 Signing Assertions

1804 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as
1805 described in Section 2.

1805 5.2 Request/Response Signing

1806 All SAML protocol request and response messages MAY be signed using XML Signature. This is
1807 reflected in the schema as described in Section 3.

1807 5.3 Signature Inheritance

1808 A SAML assertion may be embedded within another SAML element, such as an enclosing
1809 <Assertion> or a request or response, which may be signed. When a SAML assertion does not contain
1810 a <ds:Signature> element, but is contained in an enclosing SAML element that contains a

1809 <ds:Signature> element, and the signature applies to the <Assertion> element and all its children,
1810 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
1811 interpretation should be equivalent to the case where the assertion itself was signed with the same key
1812 and signature options.

1810 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
1811 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY
1812 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
1813 information from the surrounding context, but no such inheritance should be inferred unless specifically
1814 identified by the profile.

1811 5.4 XML Signature Profile

1812 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1813 and many choices. This section details constraints on these facilities so that SAML processors do not
1814 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
1815 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**
1816 attribute on <Assertion> and the various request and response elements. These attributes are
1817 collectively referred to in this section as the identifier attributes.

1813 Note that this profile only applies to the use of the <ds:Signature> elements found directly within
1814 SAML assertions, requests, and responses. Other profiles in which signatures appear elsewhere but
1815 apply to SAML content are free to define other approaches.

1814 5.4.1 Signing Formats and Algorithms

1815 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1816 detached.

1816 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
1817 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
1818 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

1817 5.4.2 References

1818 SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of
1819 the assertion or protocol message being signed. The assertion's or protocol message's root element may
1820 or may not be the root element of the actual XML document containing the signed assertion or protocol
1821 message (e.g., it might be contained within a SOAP envelope).

1819 Signatures MUST contain a single <ds:Reference> containing a same-document reference to the **ID**
1820 attribute value of the root element of the assertion or protocol message being signed. For example, if the
1821 **ID** attribute value is "foo", then the **URI** attribute in the <ds:Reference> element MUST be "#foo".

1820 5.4.3 Canonicalization Method

1821 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without
1822 comments, both in the <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a
1823 <ds:Transform> algorithm. Use of Exclusive Canonicalization ensures that signatures created over
1824 SAML messages embedded in an XML context can be verified independent of that context.

1822 5.4.4 Transforms

1823 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
1824 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
1825 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1826 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1824 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1825 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This
1826 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
1827 applying the transforms manually to the content and reverifying the result as consisting of the same
1828 SAML message.

1825 5.4.5 KeyInfo

1826 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of
1827 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be
1828 absent.

1827 5.4.6 Example

1828 Following is an example of a signed response containing a signed assertion. Line breaks have been
1829 added for readability; the signatures are not valid and cannot be successfully verified.

```
1829 <Response
1830   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
1831   ID="_c7055387-af61-4fce-8b98-e2927324b306"
1832   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
1833   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
1834   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
1835   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1836     <ds:SignedInfo>
1837       <ds:CanonicalizationMethod
1838         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1839       <ds:SignatureMethod
1840         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1841       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">
1842         <ds:Transforms>
1843           <ds:Transform
1844             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
1845 signature" />
1846           <ds:Transform
1847             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1848             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
1849             xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1850           </ds:Transform>
1851         </ds:Transforms>
1852         <ds:DigestMethod
1853           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1854         <ds:DigestValue>TCDVSVuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
1855       </ds:Reference>
1856     </ds:SignedInfo>
1857     <ds:SignatureValue>
1858       x/GyPbzmfEe85pGD3c1aXG4Vs9V9jGcJwCRKrtwPS6vdVNCcY5rHaFPYwKf+5
1859       EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
1860       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
1861     </ds:SignatureValue>
1862     <ds:KeyInfo>
1863       <ds:X509Data>
1864         <ds:X509Certificate>
```

1864
1865
1866
1866
1867
1867
1868
1868
1869
1869
1870
1870
1871
1871
1872
1872
1873
1873
1874
1874
1875
1875
1876
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1910
1911
1912

```
MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgAkxCzAJBgNVBAYTA1VT
MRlWwEAYDVQQQIEw1XaXNjb25zaW4xEDA0BgNVBACTB01hZGlzb24xIDAeBgNVBAoT
F1VuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBj
bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsX
CzAJBgNVBAYTA1VTMREwDwYDVQQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
Ym9yMQ4wDAYDVQQKEwVWQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJ1ZXQyLmVh
dTenMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbmRlcm5ldDIuZWZ1ZWR1MIGfMAOG
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIaOAPSZB113R6+KYiE7x4XAWIRCP+
c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
qgi7lFV6MDkhhTvtqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
<Status>
  <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</Status>
<Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
  IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
  xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
  <Issuer>https://www.opensaml.org/IDP</Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <ds:SignatureMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
        <ds:Transforms>
          <ds:Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
          <ds:Transform
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <InclusiveNamespaces
            PrefixList="#default saml ds xs xsi"
            xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
  </ds:SignatureValue>
```

```

1913 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgzPkJN9CMLG8ENR4Nrw+n
1914
1915 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
1915 MwuL/cBUj20tBZOQMFN7jQ9YB7klIz3RqVL+wNmeWI4=
1916 </ds:SignatureValue>
1917 <ds:KeyInfo>
1918 <ds:X509Data>
1919 <ds:X509Certificate>
1920
1921 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
1921 MRlW EAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
1922 F1VuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLLEyJEaXZpc2lvbiBvZiBJ
1923 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
1924 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsX
1925 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGAlUEBxMJQW5uIEFy
1926 Ym9yMQ4wDAYDVQQKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVh
1927 dTEhMCUGCSqGSIb3DQEJARYYcm9vdEBzaGlMS5pbNlcm5ldDIuZWZWR1MIGfMAOG
1928 CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sXvhaXnXVIVT8vuRay+x50z7GJj
1929 IHRyQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB113R6+KYIE7x4XAWIRCP+
1930 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
1931 pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
1932 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
1933 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
1934 8I3bsbmRAUg4UP9hH6ABVq4KQKmnxulxQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
1935 </ds:X509Certificate>
1936 </ds:X509Data>
1937 </ds:KeyInfo>
1938 </ds:Signature>
1939 <Subject>
1940 <NameID
1941 Format="urn:oasis:names:tc:SAML:1.1:nameid-
1942 format:emailAddress">
1943 scott@example.org
1944 </NameID>
1945 <SubjectConfirmation
1946 Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
1947 </Subject>
1948 <Conditions NotBefore="2003-04-17T00:46:02Z"
1949 NotOnOrAfter="2003-04-17T00:51:02Z">
1950 <AudienceRestriction>
1951 <Audience>http://www.opensaml.org/SP</Audience>
1952 </AudienceRestriction>
1953 </Conditions>
1954 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
1955 <AuthnContext>
1956 <AuthnContextClassRef>
1957 urn:oasis:names:tc:SAML:2.0:ac:classes:Password
1958 </AuthnContextClassRef>
1959 </AuthnContext>
1960 </AuthnStatement>
</Assertion>

```

1961

</Response>

1962 6 SAML and XML Encryption Syntax and Processing

1963 Encryption is used as the means to implement confidentiality. The most common motives for
1964 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for
1965 competitive advantage or similar reasons. Confidentiality may also be required to ensure the
1966 effectiveness of some other security mechanism. For example, a secret password or key may be
1967 encrypted.

1964 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 1965 • Communications confidentiality may be provided by mechanisms associated with a particular
1966 binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS (see
1967 [RFC 2246]/[SSL3]) or SOAP Message Security mechanisms for confidentiality.
- 1966 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`
1967 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be
1968 encrypted.
- 1967 • An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- 1968 • The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- 1969 • An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

1970 6.1 General Considerations

1971 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by
1972 use of XML Encryption [XMLEnc]. Encrypted data and optionally [PE30] zero or more encrypted keys
1973 MUST replace the plaintext information in the same location within the XML instance. The
1974 `<EncryptedData>` element's `Type` attribute SHOULD be used and, if it is present, MUST have the
1975 value `http://www.w3.org/2001/04/xmlenc#Element`.

1972 Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The
1973 SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

1973 6.2 ~~Combining Signatures and Encryption~~[PE43] Key and Data 1974 Referencing Guidelines

1974 ~~Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and~~
1975 ~~encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in~~
1976 ~~the reverse order that signing and encryption were performed.~~

- 1975 • ~~When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and~~
1976 ~~placed within the `<Assertion>` element before the element is encrypted.~~
- 1976 • ~~When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be~~
1977 ~~performed first and then the signature calculated over the assertion or message containing the~~
1978 ~~encrypted element.~~

1977 ~~If an encrypted key is NOT included in the XML instance, then the relying party must be able to locally~~
1978 ~~determine the decryption key, per [XMLEnc].~~

1978 ~~Implementations of SAML MAY implicitly associate keys with the corresponding data they are used to~~
1979 ~~encrypt, through the positioning of `<xenc:EncryptedKey>` elements next to the associated~~
1980 ~~`<xenc:EncryptedData>` element, within the enclosing SAML parent element. However, the following~~
1981 ~~set of explicit referencing guidelines are suggested to facilitate interoperability.~~

1979 [If the encrypted key is included in the XML instance, then it SHOULD be referenced within the](#)
1980 [associated <xenc:EncryptedData> element, or alternatively embedded within the](#)
1981 [<xenc:EncryptedData> element. When an <xenc:EncryptedKey> element is used, the](#)
1982 [<ds:KeyInfo> element within <xenc:EncryptedData> SHOULD reference the](#)
1983 [<xenc:EncryptedKey> element using a <ds:RetrievalMethod> element of Type](#)
1984 <http://www.w3.org/2001/04/xmlenc#EncryptedKey>.

1980 [In addition, an <xenc:EncryptedKey> element SHOULD contain an <xenc:ReferenceList>](#)
1981 [element containing a <xenc:DataReference> that references the corresponding](#)
1982 [<xenc:EncryptedData> element\(s\) that the key was used to encrypt.](#)

1981 [In scenarios where the encrypted element is being "multicast" to multiple recipients, and the key used to](#)
1982 [encrypt the message must be in turn encrypted individually and independently for each of the multiple](#)
1983 [recipients, the <xenc:CarriedKeyName> element SHOULD be used to assign a common name to](#)
1984 [each of the <xenc:EncryptedKey> elements so that a <ds:KeyName> can be used from within the](#)
1985 [<xenc:EncryptedData> element's <ds:KeyInfo> element.](#)

1982 [Within the <xenc:EncryptedData> element, the <ds:KeyName> can be thought of as an "alias" that](#)
1983 [is used for backwards referencing from the <xenc:CarriedKeyName> element in each individual](#)
1984 [<xenc:EncryptedKey> element. While this accommodates a "multicast" approach, each recipient](#)
1985 [must be able to understand \(at least one\) <ds:KeyName>. The Recipient attribute is used to provide a](#)
1986 [hint as to which key is meant for which recipient.](#)

1983 [The SAML implementation has the discretion to accept or reject a message where multiple Recipient](#)
1984 [attributes or <ds:KeyName> elements are understood. It is RECOMMENDED that implementations](#)
1985 [simply use the first key they understand and ignore any additional keys.](#)

1984 **6.3 Examples**

1985 [In the following example, the parent element \(<EncryptedID>\) contains <xenc:EncryptedData>](#)
1986 [and \(referenced\) <xenc:EncryptedKey> elements as siblings \(note that the key can in fact be](#)
1987 [anywhere in the same instance, and the key references the <xenc:EncryptedData> element\):](#)

```
1986 <saml:EncryptedID xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">  
1987   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"   
1988     Id="Encrypted_DATA_ID"  
1989     Type="http://www.w3.org/2001/04/xmlenc#Element">  
1990     <xenc:EncryptionMethod  
1991       Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>  
1992     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
1993       <ds:RetrievalMethod URI="#Encrypted_KEY_ID"  
1994         Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>  
1995     </ds:KeyInfo>  
1996     <xenc:CipherData>  
1997       <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>  
1998     </xenc:CipherData>  
1999   </xenc:EncryptedData>  
2000  
2001   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"   
2002     Id="Encrypted_KEY_ID">  
2003     <xenc:EncryptionMethod  
2004       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>  
2005     <xenc:CipherData>  
2006       <xenc:CipherValue>PzA5X...</xenc:CipherValue>  
2007     </xenc:CipherData>  
2008     <xenc:ReferenceList>  
2009       <xenc:DataReference URI="#Encrypted_DATA_ID"/>  
2010     </xenc:ReferenceList>  
2011   </xenc:EncryptedKey>
```

2012 | `</saml:EncryptedID>`

2013 | [In the following <EncryptedAttribute> example, the <xenc:EncryptedKey> element is contained](#)
2014 | [within the <xenc:EncryptedData> element, so there is no explicit referencing:](#)

```
2014 | <saml:EncryptedAttribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2015 |   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
2016 |     Id="Encrypted_DATA_ID"
2017 |     Type="http://www.w3.org/2001/04/xmlenc#Element">
2018 |     <xenc:EncryptionMethod
2019 |       Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
2020 |     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2021 |       <xenc:EncryptedKey Id="Encrypted_KEY_ID">
2022 |         <xenc:EncryptionMethod
2023 |           Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2024 |         <xenc:CipherData>
2025 |         <xenc:CipherValue>SDFSDF... </xenc:CipherValue>
2026 |         </xenc:CipherData>
2027 |       </xenc:EncryptedKey>
2028 |     </ds:KeyInfo>
2029 |     <xenc:CipherData>
2030 |     <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
2031 |     </xenc:CipherData>
2032 |   </xenc:EncryptedData>
2033 | </saml:EncryptedAttribute>
```

2034 | [The final example shows an assertion encrypted for multiple recipients, using the](#)
2035 | [<xenc:CarriedKeyName> approach:](#)

```
2035 | <saml:EncryptedAssertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2036 |   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
2037 |     Id="Encrypted_DATA_ID"
2038 |     Type="http://www.w3.org/2001/04/xmlenc#Element">
2039 |     <xenc:EncryptionMethod
2040 |       Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
2041 |     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2042 |       <ds:KeyName>MULTICAST_KEY_NAME</ds:KeyName>
2043 |     </ds:KeyInfo>
2044 |     <xenc:CipherData>
2045 |     <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
2046 |     </xenc:CipherData>
2047 |   </xenc:EncryptedData>
2048 |
2049 |   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
2050 |     Id="Encrypted_KEY_ID_1" Recipient="https://sp1.org">
2051 |     <xenc:EncryptionMethod
2052 |       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2053 |     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2054 |       <ds:KeyName>KEY_NAME_1</ds:KeyName>
2055 |     </ds:KeyInfo>
2056 |     <xenc:CipherData>
2057 |     <xenc:CipherValue>xyzABC...</xenc:CipherValue>
2058 |     </xenc:CipherData>
2059 |     <xenc:ReferenceList>
2060 |     <xenc:DataReference URI="#Encrypted_DATA_ID"/>
2061 |     </xenc:ReferenceList>
2062 |     <xenc:CarriedKeyName>MULTICAST_KEY_NAME</xenc:CarriedKeyName>
2063 |   </xenc:EncryptedKey>
2064 |
2065 |   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
2066 |     Id="Encrypted_KEY_ID_2" Recipient="https://sp2.org">
2067 |     <xenc:EncryptionMethod
2068 |       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2069 |     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2070 |       <ds:KeyName>KEY_NAME_2</ds:KeyName>
```



```
2071 | </ds:KeyInfo>
2072 | <xenc:CipherData>
2073 | <xenc:CipherValue>abcXYZ...</xenc:CipherValue>
2074 | </xenc:CipherData>
2075 | <xenc:ReferenceList>
2076 | <xenc:DataReference URI="#Encrypted_DATA_ID"/>
2077 | </xenc:ReferenceList>
2078 | <xenc:CarriedKeyName>MULTICAST_KEY_NAME</xenc:CarriedKeyName>
2079 | </xenc:EncryptedKey>
2080 | </saml:EncryptedAssertion>
```

2081 7 SAML Extensibility

2082 SAML supports extensibility in a number of ways, including extending the assertion and protocol
2083 schemas. An example of an application that extends SAML assertions is the Liberty Protocols and
2084 Schema Specification [LibertyProt]. The following sections explain the extensibility features with SAML
2085 assertions and protocols.

2083 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can
2084 be combined with extensions to put the SAML framework to new uses.

2084 7.1 Schema Extension

2085 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
2086 elements can serve as the head element of a substitution group. However, SAML types are not defined
2087 as *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that
2088 extensions are typically defined only as types rather than elements, and are included in SAML instances
2089 by means of an `xsi:type` attribute.

2086 The following sections discuss only elements and types that have been specifically designed to support
2087 extensibility.

2087 7.1.1 Assertion Schema Extension

2088 The SAML assertion schema (see [SAML-XSD]) is designed to permit separate processing of the
2089 assertion package and the statements it contains, if the extension mechanism is used for either part.

2089 The following elements are intended specifically for use as extension points in an extension schema;
2090 their types are set to *abstract*, and are thus usable only as the base of a derived type:

- 2090 • `<BaseID>` and **BaseIDAbstractType**
- 2091 • `<Condition>` and **ConditionAbstractType**
- 2092 • `<Statement>` and **StatementAbstractType**

- 2093 • The following constructs that are directly usable as part of SAML are particularly interesting targets
2094 for extension:
 - 2094 • `<AuthnStatement>` and **AuthnStatementType**
 - 2095 • `<AttributeStatement>` and **AttributeStatementType**
 - 2096 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
 - 2097 • `<AudienceRestriction>` and **AudienceRestrictionType**
 - 2098 • `<ProxyRestriction>` and **ProxyRestrictionType**
 - 2099 • `<OneTimeUse>` and **OneTimeUseType**

2100 7.1.2 Protocol Schema Extension

2101 The following SAML protocol elements are intended specifically for use as extension points in an
2102 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
2103 type:

- 2102 • `<Request>` and **RequestAbstractType**

2103 • <SubjectQuery> and **SubjectQueryAbstractType**

2104 The following constructs that are directly usable as part of SAML are particularly interesting targets for
2105 extension:

2106 • <AuthnQuery> and **AuthnQueryType**

2107 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**

2108 • <AttributeQuery> and **AttributeQueryType**

2109 • **StatusResponseType**

2109 7.2 Schema Wildcard Extension Points

2110 The SAML schemas use wildcard constructs in some locations to allow the use of elements and
2111 attributes from arbitrary namespaces, which serves as a built-in extension point without requiring an
2112 extension schema.

2111 7.2.1 Assertion Extension Points

2112 The following constructs in the assertion schema allow constructs from arbitrary namespaces within
2113 them:

2113 • <SubjectConfirmationData>: Uses **xs:anyType**, which allows any sub-elements and
2114 attributes.

2114 • <AuthnContextDecl>: Uses **xs:anyType**, which allows any sub-elements and attributes.

2115 • <AttributeValue>: Uses **xs:anyType**, which allows any sub-elements and attributes.

2116 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other
2117 namespaces with lax schema validation processing.

2117 The following constructs in the assertion schema allow arbitrary global attributes:

2118 • <Attribute> and **AttributeType**

2119 7.2.2 Protocol Extension Points

2120 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

2121 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema
2122 validation processing.

2122 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax
2123 schema validation processing.

2123 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with
2124 lax schema validation processing. (It is specifically intended to carry a SAML request or response
2125 message element, however.)

2124 7.3 Identifier Extension

2125 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
2126 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
2127 However, it is always possible to define additional URI-based identifiers for these purposes. It is

2126 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should
2127 the meaning of a given URI used as such an identifier significantly change, or be used to mean two
2128 different things.

2127 8 SAML-Defined Identifiers

2128 The following sections define URI-based identifiers for common resource access actions, subject name
2129 identifier formats, and attribute name formats.

2129 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of
2130 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
2131 have one of the following stems, according to the specification set version in which they were first
2132 introduced:

```
2130 urn:oasis:names:tc:SAML:1.0:  
2131 urn:oasis:names:tc:SAML:1.1:  
2132 urn:oasis:names:tc:SAML:2.0:
```

2133 8.1 Action Namespace Identifiers

2134 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to
2135 common sets of actions to perform on resources.

2135 8.1.1 Read/Write/Execute/Delete/Control

2136 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

2137 Defined actions:

2138 `Read Write Execute Delete Control`

2139 These actions are interpreted as follows:

2140 `Read`

2141 The subject may read the resource.

2142 `Write`

2143 The subject may modify the resource.

2144 `Execute`

2145 The subject may execute the resource.

2146 `Delete`

2147 The subject may delete the resource.

2148 `Control`

2149 The subject may specify the access control policy for the resource.

2150 8.1.2 Read/Write/Execute/Delete/Control with Negation

2151 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

2152 Defined actions:

2153 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

2154 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions
2155 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
2156 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is
2157 affirmatively denied read permission.

2155 A SAML authority MUST NOT authorize both an action and its negated form.

2156 **8.1.3 Get/Head/Put/Post**

2157 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

2158 Defined actions:

2159 GET HEAD PUT POST

2160 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
2161 the GET action on a resource is authorized to retrieve it.

2161 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and
2162 POST actions to the write permission. The correspondence is not exact however since an HTTP GET
2163 operation may cause data to be modified and a POST operation may cause modification to a resource
2164 other than the one specified in the request. For this reason a separate Action URI reference specifier is
2165 provided.

2162 **8.1.4 UNIX File Permissions**

2163 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

2164 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)
2165 notation.

2165 The action string is a four-digit numeric code:

2166 *extended user group world*

2167 Where the *extended* access permission has the value

2168 +2 if sgid is set

2169 +4 if suid is set

2170 The *user group* and *world* access permissions have the value

2171 +1 if execute permission is granted

2172 +2 if write permission is granted

2173 +4 if read permission is granted

2174 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
2175 and execute; and world read.

2175 **8.2 Attribute Name Format Identifiers**

2176 The following identifiers MAY be used in the NameFormat attribute defined on the AttributeType
2177 complex type to refer to the classification of the attribute name for purposes of interpreting the name.

2177 **8.2.1 Unspecified**

2178 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

2179 The interpretation of the attribute name is left to individual implementations.

2180 8.2.2 URI Reference

2181 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

2182 The attribute name follows the convention for URI references [RFC 2396], for example as used in
2183 XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is
2184 application-specific. See [SAMLProf] for attribute profiles that make use of this identifier.

2183 8.2.3 Basic

2184 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

2185 The class of strings acceptable as the attribute name MUST be drawn from the set of values belonging
2186 to the primitive type **xs:Name** as defined in [Schema2] Section 3.3.6. See [SAMLProf] for attribute
2187 profiles that make use of this identifier.

2186 8.3 Name Identifier Format Identifiers

2187 The following identifiers MAY be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
2188 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and
2189 the associated processing rules, if any.

2188 **Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for
2189 SAML V2.0.

2189 8.3.1 Unspecified

2190 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

2191 The interpretation of the content of the element is left to individual implementations.

2192 8.3.2 Email Address

2193 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

2194 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
2195 defined in IETF RFC 2822 [RFC 2822] Section 3.4.1. An addr-spec has the form local-part@domain.
2196 Note that an addr-spec has no phrase (such as a common name) before it, has no comment (text
2197 surrounded in parentheses) after it, and is not surrounded by "<" and ">".

2195 8.3.3 X.509 Subject Name

2196 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

2197 Indicates that the content of the element is in the form specified for the contents of the
2198 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
2199 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
2200 differ from the rules given in IETF RFC 2253 [RFC 2253].

2198 8.3.4 Windows Domain Qualified Name

2199 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

2200 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
2201 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"
2202 separator MAY be omitted.

2201 **8.3.5 Kerberos Principal Name**

2202 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

2203 Indicates that the content of the element is in the form of a Kerberos principal name using the format
2204 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
2205 realm are described in IETF RFC 1510 [RFC 1510].

2204 **8.3.6 Entity Identifier**

2205 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

2206 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
2207 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a
2208 service provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer>
2209 element to identify the issuer of a SAML request, response, or assertion, or within the <NameID>
2210 element to make assertions about system entities that can issue SAML requests, responses, and
2211 assertions. It can also be used in other elements and attributes whose purpose is to identify a system
2212 entity in various protocol exchanges.

2207 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
2208 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

2208 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

2209 **8.3.7 Persistent Identifier**

2210 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

2211 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
2212 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
2213 generated by identity providers MUST be constructed using pseudo-random values that have no
2214 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
2215 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
2216 Persistent name identifier values MUST NOT exceed a length of 256 characters.

2212 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity
2213 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived
2214 from the context of the message containing the element, such as the issuer of a protocol message or an
2215 assertion containing the identifier in its subject. Note that a different system entity might later issue its
2216 own protocol message or assertion containing the identifier; the `NameQualifier` attribute does not
2217 change in this case, but MUST continue to identify the entity that originally created the identifier (and
2218 MUST NOT be omitted in such a case).

2213 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service
2214 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be
2215 omitted if the element is contained in a message intended only for consumption directly by the service
2216 provider, and the value would be the unique identifier of that service provider.

2214 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most
2215 recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been
2216 established, then the attribute MUST be omitted.

2215 Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be
2216 shared in clear text with providers other than the providers that have established the shared identifier.
2217 Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and
2218 protections. Deployments without such requirements are free to use other kinds of identifiers in their
2219 SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

2216 Note also that while persistent identifiers are typically used to reflect an account linking relationship
2217 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
2218 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not
2219 discernibly different and does not affect the behavior of the identity provider or any processing rules
2220 specific to persistent identifiers in the protocols defined in this specification.

2217 Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of
2218 creation, but not of use. If a persistent identifier is created by a particular identity provider, the
2219 `NameQualifier` attribute value is permanently established at that time. If a service provider that
2220 receives such an identifier takes on the role of an identity provider and issues its own assertion
2221 containing that identifier, the `NameQualifier` attribute value does not change (and would of course not
2222 be omitted). It might alternatively choose to create its own persistent identifier to represent the principal
2223 and link the two values. This is a deployment decision.

2218 **8.3.8 Transient Identifier**

2219 **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

2220 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
2221 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated
2222 in accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of
2223 256 characters.

2221 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
2222 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
2223 accordance with the rules specified in Section 8.3.7.

2222 **8.4 Consent Identifiers**

2223 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType**
2224 and **StatusResponseType** complex types to communicate whether a principal gave consent, and under
2225 what conditions, for the message.

2224 **8.4.1 Unspecified**

2225 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

2226 No claim as to principal consent is being made.

2227 **8.4.2 Obtained**

2228 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

2229 Indicates that a principal's consent has been obtained by the issuer of the message.

2230 **8.4.3 Prior**

2231 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`

2232 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to
2233 the action that initiated the message.

2233 **8.4.4 Implicit**

2234 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

2235 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the
2236 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically
2237 more proximal to the action in time and presentation than prior consent, such as part of a session of
2238 activities.

2236 **8.4.5 Explicit**

2237 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

2238 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the
2239 action that initiated the message.

2239 **8.4.6 Unavailable**

2240 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

2241 Indicates that the issuer of the message did not obtain consent.

2242 **8.4.7 Inapplicable**

2243 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

2244 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

2245 9 References

2246 The following works are cited in the body of this specification.

2247 9.1 Normative References

- 2248 **[Excl-C14N]** J. Boyer et al. *Exclusive XML Canonicalization Version 1.0*. World Wide
2249 Web Consortium, July 2002. See <http://www.w3.org/TR/xml-exc-c14n/>.
- 2249 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
2250 Consortium Recommendation, May 2001. See
2251 <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively
2252 references [Schema2], listed below.
- 2250 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web
2251 Consortium Recommendation, May 2001. See
2252 <http://www.w3.org/TR/xmlschema-2/>.
- 2251 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second
2252 Edition)*. World Wide Web Consortium, October 2000. See
2253 <http://www.w3.org/TR/REC-xml>.
- 2252 **[XMLEnc]** D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide
2253 Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.
2254 Note that this specification normatively references [XMLEnc-XSD], listed below.
- 2253 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium. See
2254 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 2254 **[XMLNS]** T. Bray et al. *Namespaces in XML*. World Wide Web Consortium,
2255 January 1999. See <http://www.w3.org/TR/REC-xml-names>.
- 2255 **[XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide
2256 Web Consortium, February 2002. See <http://www.w3.org/TR/xmlsig-core/>. Note
2257 that this specification normatively references [XMLSig-XSD], listed below.
- 2256 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium. See
2257 [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)
2258 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

2257 9.2 Non-Normative References

- 2258 **[LibertyProt]** J. Beatty et al. *Liberty Protocols and Schema Specification Version 1.1*.
2259 Liberty Alliance Project, January 2003. See
2260 [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)
2261 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 2259 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.
2260 IETF RFC 1510, September 1993. See <http://www.ietf.org/rfc/rfc1510.txt>.
- 2260 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
2261 RFC 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 2261 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January
2262 1999. See <http://www.ietf.org/rfc/rfc2246.txt>.
- 2262 **[RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String
2263 Representation of Distinguished Names*. IETF RFC 2253, December 1997. See
2264 <http://www.ietf.org/rfc/rfc2253.txt>.
- 2263 **[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*.
2264 IETF RFC 2396, August, 1998. See <http://www.ietf.org/rfc/rfc2396.txt>.

2264	[RFC 2822]	P. Resnick. <i>Internet Message Format</i> . IETF RFC 2822, April 2001. See http://www.ietf.org/rfc/rfc2822.txt .
2265		
2265	[RFC 3075]	D. Eastlake, J. Reagle, D. Solo. <i>XML-Signature Syntax and Processing</i> . IETF RFC 3075, March 2001. See http://www.ietf.org/rfc/rfc3075.txt .
2266		
2266	[RFC 3513]	R. Hinden, S. Deering, <i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i> . IETF RFC 3513, April 2003. See http://www.ietf.org/rfc/rfc3513.txt .
2267		
2267	[SAMLAuthnCxt]	J. Kemp et al. <i>Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See http://www.oasis-open.org/committees/security/ .
2268		
2268	[SAMLBind]	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/ .
2269		
2269	[SAMLConform]	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. http://www.oasis-open.org/committees/security/ .
2270		
2270	[SAMLGloss]	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/ .
2271		
2271	[SAMLMeta]	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/ .
2272		
2272	[SAMLXSD]	S. Cantor et al. SAML protocols schema. OASIS SSTC, March 2005. Document ID saml-schema-protocol-2.0. See http://www.oasis-open.org/committees/security/ .
2273		
2273	[SAMLProf]	S. Cantor et al. <i>Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See http://www.oasis-open.org/committees/security/ .
2274		
2274	[SAMLSecure]	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/ .
2275		
2275	[SAMLTechOvw]	J. Hughes et al. SAML Technical Overview. OASIS, February 2005. Document ID sstc-saml-tech-overview-2.0-draft-03. See http://www.oasis-open.org/committees/security/ .
2276		
2276	[SAMLXSD]	S. Cantor et al., SAML assertions schema. OASIS SSTC, March 2005. Document ID saml-schema-assertion-2.0. See http://www.oasis-open.org/committees/security/ .
2277		
2277	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
2278		
2278	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. See http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
2279		
2279	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. See http://www.w3.org/TR/WD-charreq .
2280		
2280	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0: Normalization</i> . World Wide Web Consortium, February 2004. See http://www.w3.org/TR/charmod-norm/ .
2281		
2282		

2281 [XACML]
2282
2282 [XML-ID]
2283

eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See <http://www.oasis-open.org/committees/xacml>.

J. Marsh et al. *xml:id Version 1.0*, World Wide Web Consortium, April 2004. See <http://www.w3.org/TR/xml-id/>.

Appendix A. Acknowledgments

2284 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2285 Committee, whose voting members at the time of publication were:

- 2285 • Conor Cahill, AOL
- 2286 • John Hughes, Atos Origin
- 2287 • Hal Lockhart, BEA Systems
- 2288 • Mike Beach, Boeing
- 2289 • Rebekah Metz, Booz Allen Hamilton
- 2290 • Rick Randall, Booz Allen Hamilton
- 2291 • Ronald Jacobson, Computer Associates
- 2292 • Gavenraj Sodhi, Computer Associates
- 2293 • Thomas Wisniewski, Entrust
- 2294 • Carolina Canales-Valenzuela, Ericsson
- 2295 • Dana Kaufman, Forum Systems
- 2296 • Irving Reid, Hewlett-Packard
- 2297 • Guy Denton, IBM
- 2298 • Heather Hinton, IBM
- 2299 • Maryann Hondo, IBM
- 2300 • Michael McIntosh, IBM
- 2301 • Anthony Nadalin, IBM
- 2302 • Nick Ragouzis, Individual
- 2303 • Scott Cantor, Internet2
- 2304 • Bob Morgan, Internet2
- 2305 • Peter Davis, Neustar
- 2306 • Jeff Hodges, Neustar
- 2307 • Frederick Hirsch, Nokia
- 2308 • Senthil Sengodan, Nokia
- 2309 • Abbie Barbir, Nortel Networks
- 2310 • Scott Kiestler, Novell
- 2311 • Cameron Morris, Novell
- 2312 • Paul Madsen, NTT
- 2313 • Steve Anderson, OpenNetwork
- 2314 • Ari Kermaier, Oracle
- 2315 • Vamsi Motukuru, Oracle
- 2316 • Darren Platt, Ping Identity
- 2317 • Prateek Mishra, Principal Identity
- 2318 • Jim Lien, RSA Security
- 2319 • John Linn, RSA Security
- 2320 • Rob Philpott, RSA Security
- 2321 • Dipak Chopra, SAP
- 2322 • Jahan Moreh, Sigaba
- 2323 • Bhavna Bhatnagar, Sun Microsystems

- 2324 • Eve Maler, Sun Microsystems
- 2325 • Ronald Monzillo, Sun Microsystems
- 2326 • Emily Xu, Sun Microsystems
- 2327 • Greg Whitehead, Trustgenix

2328

2329 The editors also would like to acknowledge the following former SSTC members for their contributions to
2330 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 2330 • Stephen Farrell, Baltimore Technologies
- 2331 • David Orchard, BEA Systems
- 2332 • Krishna Sankar, Cisco Systems
- 2333 • Zahid Ahmed, CommerceOne
- 2334 • Tim Alsop, CyberSafe Limited
- 2335 • Carlisle Adams, Entrust
- 2336 • Tim Moses, Entrust
- 2337 • Nigel Edwards, Hewlett-Packard
- 2338 • Joe Pato, Hewlett-Packard
- 2339 • Bob Blakley, IBM
- 2340 • Marlena Erdos, IBM
- 2341 • Marc Chanliau, Netegrity
- 2342 • Chris McLaren, Netegrity
- 2343 • Lynne Rosenthal, NIST
- 2344 • Mark Skall, NIST
- 2345 • Charles Knouse, Oblix
- 2346 • Simon Godik, Overxeer
- 2347 • Charles Norwood, SAIC
- 2348 • Evan Prodromou, Securant
- 2349 • Robert Griffin, RSA Security (former editor)
- 2350 • Sai Allarvarpu, Sun Microsystems
- 2351 • Gary Ellison, Sun Microsystems
- 2352 • Chris Ferris, Sun Microsystems
- 2353 • Mike Myers, Traceroute Security
- 2354 • Phillip Hallam-Baker, VeriSign (former editor)
- 2355 • James Vanderbeek, Vodafone
- 2356 • Mark O'Neill, Vordel
- 2357 • Tony Palmer, Vordel

2358

2359 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2360 input to the OASIS Security Assertions Markup Language specifications:

- 2360 • Thomas Gross, IBM
- 2361 • Birgit Pfitzmann, IBM

Appendix B. Notices

2363 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2364 might be claimed to pertain to the implementation or use of the technology described in this document or
2365 the extent to which any license under such rights might or might not be available; neither does it
2366 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
2367 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
2368 made available for publication and any assurances of licenses to be made available, or the result of an
2369 attempt made to obtain a general license or permission for the use of such proprietary rights by
2370 implementors or users of this specification, can be obtained from the OASIS Executive Director.

2364 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
2365 or other proprietary rights which may cover technology that may be required to implement this
2366 specification. Please address the information to the OASIS Executive Director.

2365 **Copyright © OASIS Open 2005. All Rights Reserved.**

2366 This document and translations of it may be copied and furnished to others, and derivative works that
2367 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
2368 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright
2369 notice and this paragraph are included on all such copies and derivative works. However, this document
2370 itself may not be modified in any way, such as by removing the copyright notice or references to OASIS,
2371 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
2372 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required
2373 to translate it into languages other than English.

2367 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2368 or assigns.

2368 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2369 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2370 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS
2371 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
2372 PURPOSE.