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

Working Draft 06, 423 ~~February~~December 20097

Document identifier:

sstc-saml-core-errata-2.0-wd-06

Location:

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

Editors:

Scott Cantor, Internet2
John Kemp, Nokia
Rob Philpott, RSA Security
Eve Maler, Individual (errata editor)

Contributors to the Errata:

Rob Philpott, EMC Corporation
Nick Ragouzis, Enosis Group
Thomas Wisniewski, Entrust
Greg Whitehead, HP
Heather Hinton, IBM
Connor P. Cahill, Intel
Scott Cantor, Internet2
Nate Klingenstein, Internet2
RL 'Bob' Morgan, Internet2
John Bradley, Individual
Jeff Hodges, Individual
Joni Brennan, Liberty Alliance
Eric Tiffany, Liberty Alliance
Thomas Hardjono, M.I.T.
Tom Scavo, NCSA
Peter Davis, NeuStar, Inc.
Frederick Hirsch, Nokia Corporation
Paul Madsen, NTT Corporation
Ari Kermaier, Oracle Corporation
Hal Lockhart, Oracle Corporation
Prateek Mishra, Oracle Corporation
Brian Campbell, Ping Identity
Anil Saldhana, Red Hat Inc.
Jim Lien, RSA Security
Jahan Moreh, Sigaba
Kent Spaulding, Skyworth TTG Holdings Limited
Emily Xu, Sun Microsystems
David Staggs, Veteran's Health Administration

SAML V2.0 Contributors:

Conor P. Cahill, AOL

46 John Hughes, Atos Origin
47 Hal Lockhart, BEA Systems
48 Michael Beach, Boeing
49 Rebekah Metz, Booz Allen Hamilton
50 Rick Randall, Booz Allen Hamilton
51 Thomas Wisniewski, Entrust
52 Irving Reid, Hewlett-Packard
53 Paula Austel, IBM
54 Maryann Hondo, IBM
55 Michael McIntosh, IBM
56 Tony Nadalin, IBM
57 Nick Ragouzis, Individual
58 Scott Cantor, Internet2
59 RL 'Bob' Morgan, Internet2
60 Peter C Davis, Neustar
61 Jeff Hodges, Neustar
62 Frederick Hirsch, Nokia
63 John Kemp, Nokia
64 Paul Madsen, NTT
65 Steve Anderson, OpenNetwork
66 Prateek Mishra, Principal Identity
67 John Linn, RSA Security
68 Rob Philpott, RSA Security
69 Jahan Moreh, Sigaba
70 Anne Anderson, Sun Microsystems
71 Eve Maler, Sun Microsystems
72 Ron Monzillo, Sun Microsystems
73 Greg Whitehead, Trustgenix

74 **Abstract:**

75 The SAML V2.0 Assertions and Protocols specification defines the syntax and semantics for XML-
76 encoded assertions about authentication, attributes, and authorization, and for the protocols that
77 convey this information. This document, known as an “errata composite”, combines corrections to
78 reported errata with the original specification text. By design, the corrections are limited to
79 clarifications of ambiguous or conflicting specification text. This document shows deletions from
80 the original specification as struck-through text, and additions as colored underlined text. The
81 “[*Enn*]” designations embedded in the text refer to particular errata and their dispositions.

82 **Status:**

83 This errata composite document is a **working draft** based on the [original](#) OASIS Standard
84 document that had been produced by the Security Services Technical Committee and approved
85 by the OASIS membership on 1 March 2005. While the errata corrections appearing here are
86 non-normative, they reflect changes specified by the Approved Errata document (currently at
87 Working Draft revision 02), which is on an OASIS standardization track. In case of any
88 discrepancy between this document and the Approved Errata, the latter has precedence. ~~See also~~
89 ~~the Errata Working Document (currently at revision 39), which provides background on the~~
90 ~~changes specified here.~~

91 This document includes corrections for errata E0, E6, E8, E10, E12, E13, E14, E15, E30, E36,
92 E38, E43, E45, E46, E47, E49, E55, E60, E61, E65, E74, E75, E78, and E79.

93 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
94 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by following the instructions at
95 http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

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

Table of Contents

100

101	1 Introduction.....	7
102	1.1 Notation.....	7
103	1.2 Schema Organization and Namespaces.....	8
104	1.3 Common Data Types.....	8
105	1.3.1 String Values.....	8
106	1.3.2 URI Values.....	9
107	1.3.3 Time Values.....	9
108	1.3.4 ID and ID Reference Values.....	9
109	2 SAML Assertions.....	11
110	2.1 Schema Header and Namespace Declarations.....	11
111	2.2 Name Identifiers.....	12
112	2.2.1 Element <BaseID>.....	12
113	2.2.2 Complex Type NameIDType.....	13
114	2.2.3 Element <NameID>.....	14
115	2.2.4 Element <EncryptedID>.....	14
116	2.2.5 Element <Issuer>.....	15
117	2.3 Assertions.....	15
118	2.3.1 Element <AssertionIDRef>.....	15
119	2.3.2 Element <AssertionURIRef>.....	15
120	2.3.3 Element <Assertion>.....	15
121	2.3.4 Element <EncryptedAssertion>.....	17
122	2.4 Subjects.....	18
123	2.4.1 Element <Subject>.....	18
124	2.4.1.1 Element <SubjectConfirmation>.....	18
125	2.4.1.2 Element <SubjectConfirmationData>.....	19
126	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	21
127	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
128	2.5 Conditions.....	21
129	2.5.1 Element <Conditions>.....	22
130	2.5.1.1 General Processing Rules.....	22
131	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
132	2.5.1.3 Element <Condition>.....	23
133	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	24
134	2.5.1.5 Element <OneTimeUse>.....	24
135	2.5.1.6 Element <ProxyRestriction>.....	25
136	2.6 Advice.....	26
137	2.6.1 Element <Advice>.....	26
138	2.7 Statements.....	27
139	2.7.1 Element <Statement>.....	27
140	2.7.2 Element <AuthnStatement>.....	27
141	2.7.2.1 Element <SubjectLocality>.....	28
142	2.7.2.2 Element <AuthnContext>.....	29
143	2.7.3 Element <AttributeStatement>.....	29
144	2.7.3.1 Element <Attribute>.....	30
145	2.7.3.1.1 Element <AttributeValue>.....	31
146	2.7.3.2 Element <EncryptedAttribute>.....	32
147	2.7.4 Element <AuthzDecisionStatement>.....	32
148	2.7.4.1 Simple Type DecisionType.....	33
149	2.7.4.2 Element <Action>.....	34

150	2.7.4.3 Element <Evidence>.....	34
151	3 SAML Protocols.....	36
152	3.1 Schema Header and Namespace Declarations.....	36
153	3.2 Requests and Responses.....	37
154	3.2.1 Complex Type RequestAbstractType.....	37
155	3.2.2 Complex Type StatusResponseType.....	39
156	3.2.2.1 Element <Status>.....	40
157	3.2.2.2 Element <StatusCode>.....	41
158	3.2.2.3 Element <StatusMessage>.....	43
159	3.2.2.4 Element <StatusDetail>.....	43
160	3.3 Assertion Query and Request Protocol.....	43
161	3.3.1 Element <AssertionIDRequest>.....	43
162	3.3.2 Queries.....	44
163	3.3.2.1 Element <SubjectQuery>.....	44
164	3.3.2.2 Element <AuthnQuery>.....	44
165	3.3.2.2.1 Element <RequestedAuthnContext>.....	45
166	3.3.2.3 Element <AttributeQuery>.....	46
167	3.3.2.4 Element <AuthzDecisionQuery>.....	47
168	3.3.3 Element <Response>.....	48
169	3.3.4 Processing Rules.....	48
170	3.4 Authentication Request Protocol.....	49
171	3.4.1 Element <AuthnRequest>.....	49
172	3.4.1.1 Element <NameIDPolicy>.....	52
173	3.4.1.2 Element <Scoping>.....	54
174	3.4.1.3 Element <IDPList>.....	54
175	3.4.1.3.1 Element <IDPEntry>.....	55
176	3.4.1.4 Processing Rules.....	55
177	3.4.1.5 Proxying.....	56
178	3.4.1.5.1 Proxying Processing Rules.....	56
179	3.5 Artifact Resolution Protocol.....	58
180	3.5.1 Element <ArtifactResolve>.....	58
181	3.5.2 Element <ArtifactResponse>.....	59
182	3.5.3 Processing Rules.....	59
183	3.6 Name Identifier Management Protocol.....	60
184	3.6.1 Element <ManageNameIDRequest>.....	60
185	3.6.2 Element <ManageNameIDResponse>.....	61
186	3.6.3 Processing Rules.....	61
187	3.7 Single Logout Protocol.....	62
188	3.7.1 Element <LogoutRequest>.....	63
189	3.7.2 Element <LogoutResponse>.....	64
190	3.7.3 Processing Rules.....	64
191	3.7.3.1 Session Participant Rules.....	64
192	3.7.3.2 Session Authority Rules.....	65
193	3.8 Name Identifier Mapping Protocol.....	66
194	3.8.1 Element <NameIDMappingRequest>.....	66
195	3.8.2 Element <NameIDMappingResponse>.....	67
196	3.8.3 Processing Rules.....	67
197	4 SAML Versioning.....	69
198	4.1 SAML Specification Set Version.....	69
199	4.1.1 Schema Version.....	69
200	4.1.2 SAML Assertion Version.....	69
201	4.1.3 SAML Protocol Version.....	70
202	4.1.3.1 Request Version.....	70

203	4.1.3.2 Response Version.....	70
204	4.1.3.3 Permissible Version Combinations.....	71
205	4.2 SAML Namespace Version.....	71
206	4.2.1 Schema Evolution.....	71
207	5 SAML and XML Signature Syntax and Processing.....	72
208	5.1 Signing Assertions.....	72
209	5.2 Request/Response Signing.....	72
210	5.3 Signature Inheritance.....	72
211	5.4 XML Signature Profile.....	73
212	5.4.1 Signing Formats and Algorithms.....	73
213	5.4.2 References.....	73
214	5.4.3 Canonicalization Method.....	73
215	5.4.4 Transforms.....	74
216	5.4.5 KeyInfo.....	74
217	5.4.6 Example.....	74
218	6 SAML and XML Encryption Syntax and Processing.....	78
219	6.1 General Considerations.....	78
220	6.2 Combining Signatures and Encryption[E43] Key and Data Referencing Guidelines.....	78
221	6.3 Examples.....	79
222	7 SAML Extensibility.....	82
223	7.1 Schema Extension.....	82
224	7.1.1 Assertion Schema Extension.....	82
225	7.1.2 Protocol Schema Extension.....	82
226	7.2 Schema Wildcard Extension Points.....	83
227	7.2.1 Assertion Extension Points.....	83
228	7.2.2 Protocol Extension Points.....	83
229	7.3 Identifier Extension.....	83
230	8 SAML-Defined Identifiers.....	85
231	8.1 Action Namespace Identifiers.....	85
232	8.1.1 Read/Write/Execute/Delete/Control.....	85
233	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	85
234	8.1.3 Get/Head/Put/Post.....	86
235	8.1.4 UNIX File Permissions.....	86
236	8.2 Attribute Name Format Identifiers.....	86
237	8.2.1 Unspecified.....	86
238	8.2.2 URI Reference.....	87
239	8.2.3 Basic.....	87
240	8.3 Name Identifier Format Identifiers.....	87
241	8.3.1 Unspecified.....	87
242	8.3.2 Email Address.....	87
243	8.3.3 X.509 Subject Name.....	87
244	8.3.4 Windows Domain Qualified Name.....	87
245	8.3.5 Kerberos Principal Name.....	88
246	8.3.6 Entity Identifier.....	88
247	8.3.7 Persistent Identifier.....	88
248	8.3.8 Transient Identifier.....	89
249	8.4 Consent Identifiers.....	89
250	8.4.1 Unspecified.....	89
251	8.4.2 Obtained.....	89
252	8.4.3 Prior.....	89

253	8.4.4 Implicit.....	90
254	8.4.5 Explicit.....	90
255	8.4.6 Unavailable.....	90
256	8.4.7 Inapplicable.....	90
257	9 References.....	91
258	9.1 Normative References.....	91
259	9.2 Non-Normative References.....	91
260	Appendix A. Acknowledgments.....	94
261	Appendix B. Notices.....	96
262		

1 Introduction

263

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

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

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

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

1.1 Notation

280

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

284 Listings of SAML schemas appear like this.

285 Example code listings appear like this.

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

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

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

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

Prefix	XML Namespace	Comments
		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.

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

298 1.2 Schema Organization and Namespaces

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

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

302 The SAML request-response protocol structures are defined in a schema [SAML-PS] associated with
303 the following XML namespace:

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

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

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

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

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

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

313 1.3 Common Data Types

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

316 1.3.1 String Values

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

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

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

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

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

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

341 1.3.2 URI Values

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

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

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

351 1.3.3 Time Values

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

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

356 1.3.4 ID and ID Reference Values

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

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

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

368 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom
369 generator **MUST** be seeded with unique material in order to ensure the desired uniqueness properties
370 between different systems.

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

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

2 SAML Assertions

381

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

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

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

- 396 • **Authentication:** The assertion subject was authenticated by a particular means at a particular time.
- 397 • **Attribute:** The assertion subject is associated with the supplied attributes.
- 398 • **Authorization Decision:** A request to allow the assertion subject to access the specified resource
399 has been granted or denied [\[E13\]or is indeterminate](#).

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

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

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

2.1 Schema Header and Namespace Declarations

408

409 The following schema fragment defines the XML namespaces and other header information for the
410 assertion schema:

```
411 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
412   xmlns="http://www.w3.org/2001/XMLSchema"  
413   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
414   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
415   xmlns:xenc="http://www.w3.org/2001/04/xmenc#"  
416   elementFormDefault="unqualified"  
417   attributeFormDefault="unqualified"  
418   blockDefault="substitution"  
419   version="2.0">  
420   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
421     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-  
422 20020212/xmldsig-core-schema.xsd"/>  
423   <import namespace="http://www.w3.org/2001/04/xmenc#"  
424     schemaLocation="http://www.w3.org/TR/2002/REC-xmenc-core-  
425 20021210/xenc-schema.xsd"/>  
426   <annotation>  
427     <documentation>  
428       Document identifier: saml-schema-assertion-2.0
```

```
429         Location: http://docs.oasis-open.org/security/saml/v2.0/
430         Revision history:
431         V1.0 (November, 2002):
432             Initial Standard Schema.
433         V1.1 (September, 2003):
434             Updates within the same V1.0 namespace.
435         V2.0 (March, 2005):
436             New assertion schema for SAML V2.0 namespace.
437     </documentation>
438 </annotation>
439 ...
440 </schema>
```

441 2.2 Name Identifiers

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

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

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

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

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

461 2.2.1 Element <BaseID>

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

465 **NameQualifier** [Optional]

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

468 **SPNameQualifier** [Optional]

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

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

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

```
476 <attributeGroup name="IDNameQualifiers">  
477   <attribute name="NameQualifier" type="string" use="optional"/>  
478   <attribute name="SPNameQualifier" type="string" use="optional"/>  
479 </attributeGroup>  
480 <element name="BaseID" type="saml:BaseIDAbstractType"/>  
481 <complexType name="BaseIDAbstractType" abstract="true">  
482   <attributeGroup ref="saml:IDNameQualifiers"/>  
483 </complexType>
```

484 2.2.2 Complex Type NameIDType

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

489 NameQualifier [Optional]

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

492 SPNameQualifier [Optional]

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

496 Format [Optional]

497 A URI reference representing the classification of string-based identifier information. See Section
498 8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute
499 and their associated descriptions and processing rules. Unless otherwise specified by an element
500 based on this type, if no `Format` value is provided, then the value [\[E60\]](#)
501 [urn:oasis:names:tc:SAML:1.1:nameid-](#)
502 [format:unspecified](#)~~[urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified](#)~~
503 (see Section 8.3.1) is in effect.

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

509 SPProvidedID [Optional]

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

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

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

```
519 <complexType name="NameIDType">
```

```

520     <simpleContent>
521         <extension base="string">
522             <attributeGroup ref="saml:IDNameQualifiers"/>
523             <attribute name="Format" type="anyURI" use="optional"/>
524             <attribute name="SPProvidedID" type="string" use="optional"/>
525         </extension>
526     </simpleContent>
527 </complexType>

```

528 2.2.3 Element <NameID>

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

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

```

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

```

534 2.2.4 Element <EncryptedID>

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

538 <xenc:EncryptedData> [Required]

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

544 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

```

557 <complexType name="EncryptedElementType">
558     <sequence>
559         <element ref="xenc:EncryptedData"/>
560         <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
561     </sequence>
562 </complexType>
563 <element name="EncryptedID" type="saml:EncryptedElementType"/>

```

564 2.2.5 Element <Issuer>

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

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

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

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

572 2.3 Assertions

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

575 2.3.1 Element <AssertionIDRef>

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

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

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

582 2.3.2 Element <AssertionURIRef>

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

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

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

589 2.3.3 Element <Assertion>

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

592 `Version` [Required]

593 The version of this assertion. The identifier for the version of SAML defined in this specification is
594 "2.0". SAML versioning is discussed in Section 4.

595 `ID` [Required]

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

598 `IssueInstant` [Required]

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

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

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

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

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

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

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

617 Zero or more of the following statement elements:

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

621 <AuthnStatement>
622 An authentication statement.

623 <AuthzDecisionStatement>
624 An authorization decision statement.

625 <AttributeStatement>
626 An attribute statement.

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

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

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

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

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

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

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

```
650 <element name="Assertion" type="saml:AssertionType"/>
651 <complexType name="AssertionType">
652   <sequence>
653     <element ref="saml:Issuer"/>
654     <element ref="ds:Signature" minOccurs="0"/>
655     <element ref="saml:Subject" minOccurs="0"/>
656     <element ref="saml:Conditions" minOccurs="0"/>
657     <element ref="saml:Advice" minOccurs="0"/>
658     <choice minOccurs="0" maxOccurs="unbounded">
659       <element ref="saml:Statement"/>
660       <element ref="saml:AuthnStatement"/>
661       <element ref="saml:AuthzDecisionStatement"/>
662       <element ref="saml:AttributeStatement"/>
663     </choice>
664   </sequence>
665   <attribute name="Version" type="string" use="required"/>
666   <attribute name="ID" type="ID" use="required"/>
667   <attribute name="IssueInstant" type="dateTime" use="required"/>
668 </complexType>
```

669 2.3.4 Element <EncryptedAssertion>

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

673 <xenc:EncryptedData> [Required]

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

678 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

687 2.4 Subjects

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

689 2.4.1 Element <Subject>

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

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

694 Identifies the subject.

695 <SubjectConfirmation> [Zero or More]

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

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

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

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

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

```
708 <element name="Subject" type="saml:SubjectType"/>  
709 <complexType name="SubjectType">  
710   <choice>  
711     <sequence>  
712       <choice>  
713         <element ref="saml:BaseID"/>  
714         <element ref="saml:NameID"/>  
715         <element ref="saml:EncryptedID"/>  
716       </choice>  
717       <element ref="saml:SubjectConfirmation" minOccurs="0"  
718 maxOccurs="unbounded"/>  
719     </sequence>  
720     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>  
721   </choice>  
722 </complexType>
```

723 2.4.1.1 Element <SubjectConfirmation>

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

727 Method [Required]

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

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

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

734 <SubjectConfirmationData> [Optional]

735 Additional confirmation information to be used by a specific confirmation method. For example, typical
736 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax
737 and Processing specification [XMLSig], which identifies a cryptographic key (See also Section
738 2.4.1.3). Particular confirmation methods MAY define a schema type to describe the elements,
739 attributes, or content that may appear in the <SubjectConfirmationData> element.

740 [\[E47\] If the <SubjectConfirmation> element in an assertion subject contains an identifier the issuer](#)
741 [authorizes the attesting entity to wield the assertion on behalf of that subject. A relying party MAY apply](#)
742 [additional constraints on the use of such an assertion at its discretion, based upon the identities of both](#)
743 [the subject and the attesting entity.](#)

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

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

```
748 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
749 <complexType name="SubjectConfirmationType">  
750   <sequence>  
751     <choice minOccurs="0">  
752       <element ref="saml:BaseID"/>  
753       <element ref="saml:NameID"/>  
754       <element ref="saml:EncryptedID"/>  
755     </choice>  
756     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
757   </sequence>  
758   <attribute name="Method" type="anyURI" use="required"/>  
759 </complexType>
```

760 2.4.1.2 Element <SubjectConfirmationData>

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

767 NotBefore [Optional]

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

770 NotOnOrAfter [Optional]

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

773 Recipient [Optional]

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

777 InResponseTo [Optional]

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

781 Address [Optional]

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

788 Arbitrary attributes

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

796 Arbitrary elements

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

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

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

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

```
810 <element name="SubjectConfirmationData"  
811 type="saml:SubjectConfirmationDataType"/>  
812 <complexType name="SubjectConfirmationDataType" mixed="true">  
813 <complexContent>  
814 <restriction base="anyType">  
815 <sequence>  
816 <any namespace="##any" processContents="lax" minOccurs="0"  
817 maxOccurs="unbounded"/>  
818 </sequence>  
819 <attribute name="NotBefore" type="dateTime" use="optional"/>  
820 <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
821 <attribute name="Recipient" type="anyURI" use="optional"/>  
822 <attribute name="InResponseTo" type="NCName" use="optional"/>  
823 <attribute name="Address" type="string" use="optional"/>  
824 <anyAttribute namespace="##other" processContents="lax"/>  
825 </restriction>  
826 </complexContent>  
827 </complexType>
```

828 2.4.1.3 Complex Type `KeyInfoConfirmationDataType`

829 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`
830 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in
831 some way to authenticate an attesting entity. The particular confirmation method MUST define the exact

832 mechanism by which the confirmation data can be used. The optional attributes defined by the
833 **SubjectConfirmationDataType** complex type MAY also appear.

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

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

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

```
840 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
841   <complexContent>  
842     <restriction base="saml:SubjectConfirmationDataType">  
843       <sequence>  
844         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
845       </sequence>  
846     </restriction>  
847   </complexContent>  
848 </complexType>
```

849 2.4.1.4 Example of a Key-Confirmed <Subject>

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

```
854 <Subject>  
855   <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">  
856     scott@example.org  
857   </NameID>  
858   <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">  
859     <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
860       <ds:KeyInfo>  
861         <ds:KeyName>Scott's Key</ds:KeyName>  
862       </ds:KeyInfo>  
863     </SubjectConfirmationData>  
864   </SubjectConfirmation>  
865 </Subject>
```

866 2.5 Conditions

867 This section defines the SAML constructs that place constraints on the acceptable use of SAML
868 assertions.

869 2.5.1 Element <Conditions>

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

871 NotBefore [Optional]

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

874 NotOnOrAfter [Optional]

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

877 <Condition> [Any Number]
 878 A condition of a type defined in an extension schema. An `xsi:type` attribute MUST be used to
 879 indicate the actual condition type.

880 <AudienceRestriction> [Any Number]
 881 Specifies that the assertion is addressed to a particular audience.

882 <OneTimeUse> [Optional]
 883 Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future
 884 use. Although the schema permits multiple occurrences, there MUST be at most one instance of
 885 this element.

886 <ProxyRestriction> [Optional]
 887 Specifies limitations that the asserting party imposes on relying parties that wish to subsequently act
 888 as asserting parties themselves and issue assertions of their own on the basis of the information
 889 contained in the original assertion. Although the schema permits multiple occurrences, there MUST
 890 be at most one instance of this element.

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

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

```

897 <element name="Conditions" type="saml:ConditionsType"/>
898 <complexType name="ConditionsType">
899   <choice minOccurs="0" maxOccurs="unbounded">
900     <element ref="saml:Condition"/>
901     <element ref="saml:AudienceRestriction"/>
902     <element ref="saml:OneTimeUse"/>
903     <element ref="saml:ProxyRestriction"/>
904   </choice>
905   <attribute name="NotBefore" type="dateTime" use="optional"/>
906   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
907 </complexType>
  
```

908 2.5.1.1 General Processing Rules

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

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

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

- 916 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is
 917 considered to be **Valid** with respect to condition processing.
- 918 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the
 919 assertion is considered to be **Invalid**.
- 920 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, or if an element is
 921 encountered that is not understood, then the validity of the assertion cannot be determined and is
 922 considered to be **Indeterminate**.

923 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the
924 assertion is considered to be **Valid** with respect to condition processing.

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

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

930 2.5.1.2 Attributes NotBefore and NotOnOrAfter

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

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

936 If the value for either NotBefore or NotOnOrAfter is omitted, then it is considered unspecified. If the
937 NotBefore attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), then
938 the assertion is **Valid** with respect to conditions at any time before the time instant specified by the
939 NotOnOrAfter attribute. If the NotOnOrAfter attribute is unspecified (and if all other conditions that are
940 supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions from the time instant specified
941 by the NotBefore attribute with no expiry. If neither attribute is specified (and if any other conditions that
942 are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions at any time.

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

945 2.5.1.3 Element <Condition>

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

948 The following schema fragment defines the <Condition> element and its ConditionAbstractType
949 complex type:

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

952 2.5.1.4 Elements <AudienceRestriction> and <Audience>

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

958 <Audience>

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

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

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

970 Note that multiple <AudienceRestriction> elements MAY be included in a single assertion, and each
971 MUST be evaluated independently. The effect of this requirement and the preceding definition is that
972 within a given [\[E46\]<AudienceRestrictions>condition](#), the <Audience> [elementsaudiences](#) form
973 a disjunction (an "OR") while multiple [<AudienceRestrictions> elementselements](#) form a
974 conjunction (an "AND").

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

```
977 <element name="AudienceRestriction"  
978   type="saml:AudienceRestrictionType"/>  
979 <complexType name="AudienceRestrictionType">  
980   <complexContent>  
981     <extension base="saml:ConditionAbstractType">  
982       <sequence>  
983         <element ref="saml:Audience" maxOccurs="unbounded"/>  
984       </sequence>  
985     </extension>  
986   </complexContent>  
987 </complexType>  
988 <element name="Audience" type="anyURI"/>
```

989 **2.5.1.5 Element <OneTimeUse>**

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

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

1000 The <OneTimeUse> element indicates that the assertion SHOULD be used immediately by the relying
1001 party and MUST NOT be retained for future use. Relying parties are always free to request a fresh
1002 assertion for every use. However, implementations that choose to retain assertions for future use MUST
1003 observe the <OneTimeUse> element. This condition is independent from the NotBefore and
1004 NotOnOrAfter condition information.

1005 To support the single use constraint, a relying party should maintain a cache of the assertions it has
1006 processed containing such a condition. Whenever an assertion with this condition is processed, the cache
1007 should be checked to ensure that the same assertion has not been previously received and processed by
1008 the relying party.

1009 A SAML authority MUST NOT include more than one <OneTimeUse> element within a <Conditions>
1010 element of an assertion.

1011 For the purposes of determining the validity of the <Conditions> element, the <OneTimeUse> is
1012 considered to always be valid. That is, this condition does not affect validity but is a condition on use.

1013 The following schema fragment defines the <OneTimeUse> element and its **OneTimeUseType** complex
1014 type:

```
1015 <element name="OneTimeUse" type="saml:OneTimeUseType"/>
1016 <complexType name="OneTimeUseType">
1017   <complexContent>
1018     <extension base="saml:ConditionAbstractType"/>
1019   </complexContent>
1020 </complexType>
```

1021 **2.5.1.6 Element <ProxyRestriction>**

1022 Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as asserting
1023 parties and issue subsequent assertions of their own on the basis of the information contained in the
1024 original assertion. A relying party acting as an asserting party **MUST NOT** issue an assertion that itself
1025 violates the restrictions specified in this condition on the basis of an assertion containing such a condition.

1026 The <ProxyRestriction> element contains the following elements and attributes:

1027 **Count** [Optional]

1028 Specifies the maximum number of indirections that the asserting party permits to exist between this
1029 assertion and an assertion which has ultimately been issued on the basis of it.

1030 <Audience> [Zero or More]

1031 Specifies the set of audiences to whom the asserting party permits new assertions to be issued on
1032 the basis of this assertion.

1033 A **Count** value of zero indicates that a relying party **MUST NOT** issue an assertion to another relying party
1034 on the basis of this assertion. If greater than zero, any assertions so issued **MUST** themselves contain a
1035 <ProxyRestriction> element with a **Count** value of at most one less than this value.

1036 If no <Audience> elements are specified, then no audience restrictions are imposed on the relying
1037 parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued **MUST**
1038 themselves contain an <AudienceRestriction> element with at least one of the <Audience>
1039 elements present in the previous <ProxyRestriction> element, and no <Audience> elements
1040 present that were not in the previous <ProxyRestriction> element.

1041 A SAML authority **MUST NOT** include more than one <ProxyRestriction> element within a
1042 <Conditions> element of an assertion.

1043 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>
1044 condition is considered to always be valid. That is, this condition does not affect validity but is a condition
1045 on use.

1046 The following schema fragment defines the <ProxyRestriction> element and its
1047 **ProxyRestrictionType** complex type:

```
1048 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
1049 <complexType name="ProxyRestrictionType">
1050   <complexContent>
1051     <extension base="saml:ConditionAbstractType">
1052       <sequence>
1053         <element ref="saml:Audience" minOccurs="0"
1054 maxOccurs="unbounded"/>
1055       </sequence>
1056       <attribute name="Count" type="nonNegativeInteger" use="optional"/>
1057     </extension>
1058   </complexContent>
1059 </complexType>
```

1060 2.6 Advice

1061 This section defines the SAML constructs that contain additional information about an assertion that an
1062 asserting party wishes to provide to a relying party.

1063 2.6.1 Element <Advice>

1064 The <Advice> element contains any additional information that the SAML authority wishes to provide.
1065 This information MAY be ignored by applications without affecting either the semantics or the validity of
1066 the assertion.

1067 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,
1068 <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in
1069 other non-SAML namespaces.

1070 Following are some potential uses of the <Advice> element:

- 1071 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating
1072 the claims) or indirectly (by reference to the supporting assertions).
- 1073 • State a proof of the assertion claims.
- 1074 • Specify the timing and distribution points for updates to the assertion.

1075 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
1076 <element name="Advice" type="saml:AdviceType"/>  
1077 <complexType name="AdviceType">  
1078   <choice minOccurs="0" maxOccurs="unbounded">  
1079     <element ref="saml:AssertionIDRef"/>  
1080     <element ref="saml:AssertionURIRef"/>  
1081     <element ref="saml:Assertion"/>  
1082     <element ref="saml:EncryptedAssertion"/>  
1083     <any namespace="##other" processContents="lax"/>  
1084   </choice>  
1085 </complexType>
```

1086 2.7 Statements

1087 The following sections define the SAML constructs that contain statement information.

1088 2.7.1 Element <Statement>

1089 The <Statement> element is an extension point that allows other assertion-based applications to reuse
1090 the SAML assertion framework. SAML itself derives its core statements from this extension point. Its
1091 **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

1092 The following schema fragment defines the <Statement> element and its **StatementAbstractType**
1093 complex type:

```
1094 <element name="Statement" type="saml:StatementAbstractType"/>  
1095 <complexType name="StatementAbstractType" abstract="true"/>
```

1096 2.7.2 Element <AuthnStatement>

1097 The <AuthnStatement> element describes a statement by the SAML authority asserting that the
1098 assertion subject was authenticated by a particular means at a particular time. Assertions containing
1099 <AuthnStatement> elements MUST contain a <Subject> element.

1100 It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the
1101 following elements and attributes:

1102 **Note:** The `<AuthorityBinding>` element and its corresponding type were removed
1103 from `<AuthnStatement>` for V2.0 of SAML.

1104 `AuthnInstant` [Required]

1105 Specifies the time at which the authentication took place. The time value is encoded in UTC, as
1106 described in Section 1.3.3.

1107 `SessionIndex` [Optional]

1108 Specifies the index of a particular session between the principal identified by the subject and the
1109 authenticating authority.

1110 `SessionNotOnOrAfter` [Optional]

1111 ~~Specifies a time instant at which the session between the principal identified by the subject and the~~
1112 ~~SAML authority issuing this statement MUST be considered ended.~~^[E79]~~Indicates an upper bound on~~
1113 ~~sessions with the subject derived from the enclosing assertion.~~ The time value is encoded in UTC, as
1114 described in Section 1.3.3. There is no required relationship between this attribute and a
1115 `NotOnOrAfter` condition attribute that may be present in the assertion. It's left to profiles to provide
1116 specific processing rules for relying parties based on this attribute.

1117 `<SubjectLocality>` [Optional]

1118 Specifies the DNS domain name and IP address for the system from which the assertion subject was
1119 apparently authenticated.

1120 `<AuthnContext>` [Required]

1121 The context used by the authenticating authority up to and including the authentication event that
1122 yielded this statement. Contains an authentication context class reference, an authentication context
1123 declaration or declaration reference, or both. See the Authentication Context specification
1124 [SAMLAuthnCxt] for a full description of authentication context information.

1125 In general, any string value MAY be used as a `SessionIndex` value. However, when privacy is a
1126 consideration, care must be taken to ensure that the `SessionIndex` value does not invalidate other
1127 privacy mechanisms. Accordingly, the value SHOULD NOT be usable to correlate activity by a principal
1128 across different session participants. Two solutions that achieve this goal are provided below and are
1129 RECOMMENDED:

1130 • Use small positive integers (or reoccurring constants in a list) for the `SessionIndex`. The SAML
1131 authority SHOULD choose the range of values such that the cardinality of any one integer will be
1132 sufficiently high to prevent a particular principal's actions from being correlated across multiple session
1133 participants. The SAML authority SHOULD choose values for `SessionIndex` randomly from within
1134 this range (except when required to ensure unique values for subsequent statements given to the
1135 same session participant but as part of a distinct session).

1136 • Use the enclosing assertion's ID value in the `SessionIndex`.

1137 The following schema fragment defines the `<AuthnStatement>` element and its **AuthnStatementType**
1138 complex type:

```
1139 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
1140 <complexType name="AuthnStatementType">
1141   <complexContent>
1142     <extension base="saml:StatementAbstractType">
1143       <sequence>
1144         <element ref="saml:SubjectLocality" minOccurs="0"/>
1145         <element ref="saml:AuthnContext"/>
1146       </sequence>
```

```

1147         <attribute name="AuthnInstant" type="dateTime" use="required"/>
1148         <attribute name="SessionIndex" type="string" use="optional"/>
1149         <attribute name="SessionNotOnOrAfter" type="dateTime"
1150 use="optional"/>
1151     </extension>
1152 </complexContent>
1153 </complexType>

```

1154 2.7.2.1 Element <SubjectLocality>

1155 The <SubjectLocality> element specifies the DNS domain name and IP address for the system from
1156 which the assertion subject was authenticated. It has the following attributes:

1157 Address [Optional]

1158 The network address of the system from which the principal identified by the subject was
1159 authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "1.2.3.4").
1160 IPv6 addresses SHOULD be represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513]
1161 (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

1162 DNSName [Optional]

1163 The DNS name of the system from which the principal identified by the subject was authenticated.

1164 This element is entirely advisory, since both of these fields are quite easily "spoofed," but may be useful
1165 information in some applications.

1166 The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**
1167 complex type:

```

1168 <element name="SubjectLocality" type="saml:SubjectLocalityType"/>
1169 <complexType name="SubjectLocalityType">
1170     <attribute name="Address" type="string" use="optional"/>
1171     <attribute name="DNSName" type="string" use="optional"/>
1172 </complexType>

```

1173 2.7.2.2 Element <AuthnContext>

1174 The <AuthnContext> element specifies the context of an authentication event. The element can contain
1175 an authentication context class reference, an authentication context declaration or declaration reference,
1176 or both. Its complex **AuthnContextType** has the following elements:

1177 <AuthnContextClassRef> [Optional]

1178 A URI reference identifying an authentication context class that describes the authentication context
1179 declaration that follows.

1180 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

1181 Either an authentication context declaration provided by value, or a URI reference that identifies such
1182 a declaration. The URI reference MAY directly resolve into an XML document containing the
1183 referenced declaration.

1184 <AuthenticatingAuthority> [Zero or More]

1185 Zero or more unique identifiers of authentication authorities that were involved in the authentication of
1186 the principal (not including the assertion issuer, who is presumed to have been involved without being
1187 explicitly named here).

1188 See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication
1189 context information.

1190 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**
1191 complex type:

```
1192 <element name="AuthnContext" type="saml:AuthnContextType"/>
1193 <complexType name="AuthnContextType">
1194   <sequence>
1195     <choice>
1196       <sequence>
1197         <element ref="saml:AuthnContextClassRef"/>
1198         <choice minOccurs="0">
1199           <element ref="saml:AuthnContextDecl"/>
1200           <element ref="saml:AuthnContextDeclRef"/>
1201         </choice>
1202       </sequence>
1203     <choice>
1204       <element ref="saml:AuthnContextDecl"/>
1205       <element ref="saml:AuthnContextDeclRef"/>
1206     </choice>
1207   </choice>
1208   <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1209   maxOccurs="unbounded"/>
1210 </sequence>
1211 </complexType>
1212 <element name="AuthnContextClassRef" type="anyURI"/>
1213 <element name="AuthnContextDeclRef" type="anyURI"/>
1214 <element name="AuthnContextDecl" type="anyType"/>
1215 <element name="AuthenticatingAuthority" type="anyURI"/>
```

1216 2.7.3 Element <AttributeStatement>

1217 The <AttributeStatement> element describes a statement by the SAML authority asserting that the
1218 assertion subject is associated with the specified attributes. Assertions containing
1219 <AttributeStatement> elements MUST contain a <Subject> element.

1220 It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the
1221 following elements:

1222 <Attribute> or <EncryptedAttribute> [One or More]

1223 The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML
1224 attribute may be included with the <EncryptedAttribute> element.

1225 The following schema fragment defines the <AttributeStatement> element and its
1226 **AttributeStatementType** complex type:

```
1227 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1228 <complexType name="AttributeStatementType">
1229   <complexContent>
1230     <extension base="saml:StatementAbstractType">
1231       <choice maxOccurs="unbounded">
1232         <element ref="saml:Attribute"/>
1233         <element ref="saml:EncryptedAttribute"/>
1234       </choice>
1235     </extension>
1236   </complexContent>
1237 </complexType>
```

1238 2.7.3.1 Element <Attribute>

1239 The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the
1240 **AttributeType** complex type. It is used within an attribute statement to express particular attributes and
1241 values associated with an assertion subject, as described in the previous section. It is also used in an

1242 attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.3 for
1243 more information). The <Attribute> element contains the following XML attributes:

1244 Name [Required]

1245 The name of the attribute.

1246 NameFormat [Optional]

1247 A URI reference representing the classification of the attribute name for purposes of interpreting the
1248 name. See Section 8.2 for some URI references that MAY be used as the value of the NameFormat
1249 attribute and their associated descriptions and processing rules. If no NameFormat value is provided,
1250 the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section
1251 8.2.1) is in effect.

1252 FriendlyName [Optional]

1253 A string that provides a more human-readable form of the attribute's name, which may be useful in
1254 cases in which the actual Name is complex or opaque, such as an OID or a UUID. This attribute's
1255 value MUST NOT be used as a basis for formally identifying SAML attributes.

1256 Arbitrary attributes

1257 This complex type uses an <xs:anyAttribute> extension point to allow arbitrary XML attributes to
1258 be added to <Attribute> constructs without the need for an explicit schema extension. This allows
1259 additional fields to be added as needed to supply additional parameters to be used, for example, in an
1260 attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or
1261 XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a
1262 derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1263 <AttributeValue> [Any Number]

1264 Contains a value of the attribute. If an attribute contains more than one discrete value, it is
1265 RECOMMENDED that each value appear in its own <AttributeValue> element. If more than
1266 one <AttributeValue> element is supplied for an attribute, and any of the elements have a
1267 datatype assigned through `xsi:type`, then all of the <AttributeValue> elements must have
1268 the identical datatype assigned.

1269 [\[E49\]Attributes are identified/named by the combination of the NameFormat and Name XML](#)
1270 [attributes described above. Neither one in isolation can be assumed to be unique, but taken](#)
1271 [together, they ought to be unambiguous within a given deployment.](#)

1272 [The SAML profiles specification \[SAMLProf\] includes a number of attribute profiles designed to](#)
1273 [improve the interoperability of attribute usage in some identified scenarios. Such profiles typically](#)
1274 [include constraints on attribute naming and value syntax. There is no explicit indicator when an](#)
1275 [attribute profile is in use, and it is assumed that deployments can establish this out of band, based](#)
1276 [on the combination of NameFormat and Name.](#)

1277 The meaning of an <Attribute> element that contains no <AttributeValue> elements depends on
1278 its context. Within an <AttributeStatement>, if the SAML attribute exists but has no values, then the
1279 <AttributeValue> element MUST be omitted. Within a <samlp:AttributeQuery>, the absence of
1280 values indicates that the requester is interested in any or all of the named attribute's values (see also
1281 Section 3.3.2.3).

1282 Any other uses of the <Attribute> element by profiles or other specifications MUST define the
1283 semantics of specifying or omitting <AttributeValue> elements.

1284 The following schema fragment defines the <Attribute> element and its **AttributeType** complex type:

```
1285 <element name="Attribute" type="saml:AttributeType"/>  
1286 <complexType name="AttributeType">  
1287   <sequence>  
1288     <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
```

```

1289     </sequence>
1290     <attribute name="Name" type="string" use="required"/>
1291     <attribute name="NameFormat" type="anyURI" use="optional"/>
1292     <attribute name="FriendlyName" type="string" use="optional"/>
1293     <anyAttribute namespace="##other" processContents="lax"/>
1294 </complexType>

```

1295 2.7.3.1.1 Element <AttributeValue>

1296 The <AttributeValue> element supplies the value of a specified SAML attribute. It is of the
 1297 **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

1298 If the data content of an <AttributeValue> element is of an XML Schema simple type (such as
 1299 **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration
 1300 in the <AttributeValue> element. If the attribute value contains structured data, the necessary data
 1301 elements MAY be defined in an extension schema.

1302 **Note:** Specifying a datatype other than an XML Schema simple type on
 1303 <AttributeValue> using `xsi:type` will require the presence of the extension schema
 1304 that defines the datatype in order for schema processing to proceed.

1305 If a SAML attribute includes an empty value, such as the empty string, the corresponding
 1306 <AttributeValue> element MUST be empty (generally this is serialized as <AttributeValue/>).
 1307 This overrides the requirement in Section 1.3.1 that string values in SAML content contain at least one
 1308 non-whitespace character.

1309 If a SAML attribute includes a "null" value, the corresponding <AttributeValue> element MUST be
 1310 empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "true" or "1".

1311 The following schema fragment defines the <AttributeValue> element:

```

1312 <element name="AttributeValue" type="anyType" nillable="true"/>

```

1313 2.7.3.2 Element <EncryptedAttribute>

1314 The <EncryptedAttribute> element represents a SAML attribute in encrypted fashion, as defined by
 1315 the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAttribute>
 1316 element contains the following elements:

1317 <xenc:EncryptedData> [Required]

1318 The encrypted content and associated encryption details, as defined by the XML Encryption
 1319 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
 1320 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
 1321 encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

1322 <xenc:EncryptedKey> [Zero or More]

1323 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
 1324 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
 1325 the `Recipient` attribute SHOULD be the URI identifier of a system entity with a SAML name
 1326 identifier, as defined by Section 8.3.6.

1327 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through
 1328 an intermediary.

1329 The following schema fragment defines the <EncryptedAttribute> element:

```

1330 <element name="EncryptedAttribute" type="saml:EncryptedElementType"/>

```

1331 2.7.4 Element <AuthzDecisionStatement>

1332 **Note:** The <AuthzDecisionStatement> feature has been frozen as of SAML V2.0,
1333 with no future enhancements planned. Users who require additional functionality may
1334 want to consider the eXtensible Access Control Markup Language [XACML], which offers
1335 enhanced authorization decision features.

1336 The <AuthzDecisionStatement> element describes a statement by the SAML authority asserting that
1337 a request for access by the assertion subject to the specified resource has resulted in the specified
1338 authorization decision on the basis of some optionally specified evidence. Assertions containing
1339 <AuthzDecisionStatement> elements MUST contain a <Subject> element.

1340 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1341 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
1342 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
1343 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1344 URI references are to be found in IETF RFC 2396 [RFC 2396] Section 6:

1345 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1346 dependent. When a scheme uses elements of the common syntax, it will also use the common
1347 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1348 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where
1349 the port is elided.

1350 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the
1351 URI normalized form wherever possible as follows:

- 1352 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1353 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1354 Inconsistent URI reference interpretation can also result from differences between the URI reference
1355 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1356 employed to specify an access control policy language. The following security conditions SHOULD be
1357 satisfied by the system which employs SAML assertions:

- 1358 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
1359 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
1360 part of the resource URI reference.
- 1361 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
1362 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
1363 gain access to a denied resource by creating such an equivalence.

1364 The <AuthzDecisionStatement> element is of type **AuthzDecisionStatementType**, which extends
1365 **StatementAbstractType** with the addition of the following elements and attributes:

1366 **Resource** [Required]

1367 A URI reference identifying the resource to which access authorization is sought. This attribute MAY
1368 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the
1369 current document", as specified by IETF RFC 2396 [RFC 2396] Section 4.2.

1370 **Decision** [Required]

1371 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1372 the **DecisionType** simple type.

1373 <Action> [One or more]

1374 The set of actions authorized to be performed on the specified resource.

1375 <Evidence> [Optional]

1376 A set of assertions that the SAML authority relied on in making the decision.

1377 The following schema fragment defines the <AuthzDecisionStatement> element and its
1378 **AuthzDecisionStatementType** complex type:

```
1379 <element name="AuthzDecisionStatement"  
1380 type="saml:AuthzDecisionStatementType"/>  
1381 <complexType name="AuthzDecisionStatementType">  
1382   <complexContent>  
1383     <extension base="saml:StatementAbstractType">  
1384       <sequence>  
1385         <element ref="saml:Action" maxOccurs="unbounded"/>  
1386         <element ref="saml:Evidence" minOccurs="0"/>  
1387       </sequence>  
1388       <attribute name="Resource" type="anyURI" use="required"/>  
1389       <attribute name="Decision" type="saml:DecisionType" use="required"/>  
1390     </extension>  
1391   </complexContent>  
1392 </complexType>
```

1393 2.7.4.1 Simple Type DecisionType

1394 The **DecisionType** simple type defines the possible values to be reported as the status of an
1395 authorization decision statement.

1396 Permit

1397 The specified action is permitted.

1398 Deny

1399 The specified action is denied.

1400 Indeterminate

1401 The SAML authority cannot determine whether the specified action is permitted or denied.

1402 The *Indeterminate* decision value is used in situations where the SAML authority requires the ability to
1403 provide an affirmative statement but where it is not able to issue a decision. Additional information as to
1404 the reason for the refusal or inability to provide a decision MAY be returned as <StatusDetail>
1405 elements in the enclosing <Response>.

1406 The following schema fragment defines the **DecisionType** simple type:

```
1407 <simpleType name="DecisionType">  
1408   <restriction base="string">  
1409     <enumeration value="Permit"/>  
1410     <enumeration value="Deny"/>  
1411     <enumeration value="Indeterminate"/>  
1412   </restriction>  
1413 </simpleType>
```

1414 2.7.4.2 Element <Action>

1415 The <Action> element specifies an action on the specified resource for which permission is sought. Its
1416 string-data content provides the label for an action sought to be performed on the specified resource, and
1417 it has the following attribute:

1418 Namespace [\[E36\]RequiredOptional](#)

1419 A URI reference representing the namespace in which the name of the specified action is to be
1420 interpreted. ~~If this element is absent, the namespace~~

1421 | <urn:oasis:names:tc:SAML:1.0:action:rwcde-negation> specified in Section 8.1.2 is in
1422 | effect.

1423 | The following schema fragment defines the <Action> element and its **ActionType** complex type:

```
1424 | <element name="Action" type="saml:ActionType"/>  
1425 | <complexType name="ActionType">  
1426 |   <simpleContent>  
1427 |     <extension base="string">  
1428 |       <attribute name="Namespace" type="anyURI" use="required"/>  
1429 |     </extension>  
1430 |   </simpleContent>  
1431 | </complexType>
```

1432 | 2.7.4.3 Element <Evidence>

1433 | The <Evidence> element contains one or more assertions or assertion references that the SAML
1434 | authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains
1435 | a mixture of one or more of the following elements:

1436 | <AssertionIDRef> [Any number]

1437 | Specifies an assertion by reference to the value of the assertion's ID attribute.

1438 | <AssertionURIRef> [Any number]

1439 | Specifies an assertion by means of a URI reference.

1440 | <Assertion> [Any number]

1441 | Specifies an assertion by value.

1442 | <EncryptedAssertion> [Any number]

1443 | Specifies an encrypted assertion by value.

1444 | Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1445 | and the SAML authority making the authorization decision. For example, in the case that the SAML relying
1446 | party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1447 | assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1448 | assertion as valid either to the relying party or any other third party.

1449 | The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
1450 | <element name="Evidence" type="saml:EvidenceType"/>  
1451 | <complexType name="EvidenceType">  
1452 |   <choice maxOccurs="unbounded">  
1453 |     <element ref="saml:AssertionIDRef"/>  
1454 |     <element ref="saml:AssertionURIRef"/>  
1455 |     <element ref="saml:Assertion"/>  
1456 |     <element ref="saml:EncryptedAssertion"/>  
1457 |   </choice>  
1458 | </complexType>
```

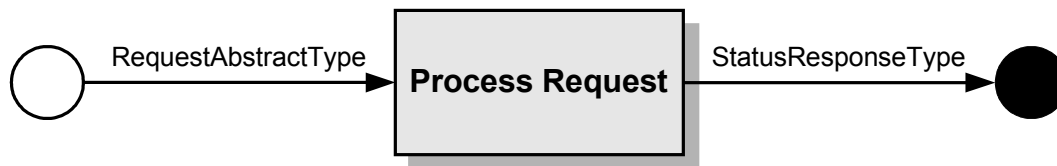
3 SAML Protocols

1459

1460 SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML
1461 bindings specification [SAMLBind] describes specific means of transporting protocol messages using
1462 existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a
1463 number of applications of the protocols defined in this section together with additional processing rules,
1464 restrictions, and requirements that facilitate interoperability.

1465 Specific SAML request and response messages derive from common types. The requester sends an
1466 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1467 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1468



1470

Figure 1: SAML Request-Response Protocol

1471 In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the
1472 responder having received a corresponding request.

1473 The protocols defined by SAML achieve the following actions:

- 1474 • Returning one or more requested assertions. This can occur in response to either a direct request
1475 for specific assertions or a query for assertions that meet particular criteria.
- 1476 • Performing authentication on request and returning the corresponding assertion
- 1477 • Registering a name identifier or terminating a name registration on request
- 1478 • Retrieving a protocol message that has been requested by means of an artifact
- 1479 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
1480 request
- 1481 • Providing a name identifier mapping on request

1482 Throughout this section, text descriptions of elements and types in the SAML protocol namespace are not
1483 shown with the conventional namespace prefix `samlp:.` For clarity, text descriptions of elements and
1484 types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:.`

3.1 Schema Header and Namespace Declarations

1485

1486 The following schema fragment defines the XML namespaces and other header information for the
1487 protocol schema:

```
1488 <schema  
1489   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1490   xmlns="http://www.w3.org/2001/XMLSchema"  
1491   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1492   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1493   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1494   elementFormDefault="unqualified"  
1495   attributeFormDefault="unqualified"  
1496   blockDefault="substitution"  
1497   version="2.0">
```

```

1498 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1499     schemaLocation="saml-schema-assertion-2.0.xsd"/>
1500 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1501     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
1502 20020212/xmldsig-core-schema.xsd"/>
1503 <annotation>
1504   <documentation>
1505     Document identifier: saml-schema-protocol-2.0
1506     Location: http://docs.oasis-open.org/security/saml/v2.0/
1507     Revision history:
1508     V1.0 (November, 2002):
1509       Initial Standard Schema.
1510     V1.1 (September, 2003):
1511       Updates within the same V1.0 namespace.
1512     V2.0 (March, 2005):
1513       New protocol schema based in a SAML V2.0 namespace.
1514   </documentation>
1515 </annotation>
1516 ...
1517 </schema>

```

1518 3.2 Requests and Responses

1519 The following sections define the SAML constructs and basic requirements that underlie all of the request
 1520 and response messages used in SAML protocols.

1521 3.2.1 Complex Type RequestAbstractType

1522 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
 1523 This type defines common attributes and elements that are associated with all SAML requests:

1524 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**
 1525 for V2.0 of SAML.

1526 ID [Required]

1527 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section
 1528 1.3.4 for identifier uniqueness. The values of the ID attribute in a request and the InResponseTo
 1529 attribute in the corresponding response MUST match.

1530 Version [Required]

1531 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".
 1532 SAML versioning is discussed in Section 4.

1533 IssueInstant [Required]

1534 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
 1535 1.3.3.

1536 Destination [Optional]

1537 A URI reference indicating the address to which this request has been sent. This is useful to prevent
 1538 malicious forwarding of requests to unintended recipients, a protection that is required by some
 1539 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
 1540 location at which the message was received. If it does not, the request MUST be discarded. Some
 1541 protocol bindings may require the use of this attribute (see [SAMLBind]).

1542 Consent [Optional]

1543 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
 1544 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value

1545 of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the
1546 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in
1547 effect.

1548 `<saml:Issuer>` [Optional]

1549 Identifies the entity that generated the request message. (For more information on this element, see
1550 Section 2.2.5.)

1551 `<ds:Signature>` [Optional]

1552 An XML Signature that authenticates the requester and provides message integrity, as described
1553 below and in Section 5.

1554 `<Extensions>` [Optional]

1555 This extension point contains optional protocol message extension elements that are agreed on
1556 between the communicating parties. No extension schema is required in order to make use of this
1557 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1558 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1559 SAML-defined namespace.

1560 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to
1561 authenticate itself, and message integrity may often be required. Authentication and message integrity
1562 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request
1563 MAY be signed, which provides both authentication of the requester and message integrity.

1564 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML
1565 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)
1566 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the
1567 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the
1568 signature to determine the identity and appropriateness of the signer and may continue to process the
1569 request or respond with an error (if the request is invalid for some other reason).

1570 If a `Consent` attribute is included and the value indicates that some form of principal consent has been
1571 obtained, then the request SHOULD be signed.

1572 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if
1573 it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value
1574 `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a
1575 suspected denial-of-service attack, not responding at all may be warranted.

1576 The following schema fragment defines the **RequestAbstractType** complex type:

```
1577 <complexType name="RequestAbstractType" abstract="true">
1578   <sequence>
1579     <element ref="saml:Issuer" minOccurs="0"/>
1580     <element ref="ds:Signature" minOccurs="0"/>
1581     <element ref="sampl:Extensions" minOccurs="0"/>
1582   </sequence>
1583   <attribute name="ID" type="ID" use="required"/>
1584   <attribute name="Version" type="string" use="required"/>
1585   <attribute name="IssueInstant" type="dateTime" use="required"/>
1586   <attribute name="Destination" type="anyURI" use="optional"/>
1587   <attribute name="Consent" type="anyURI" use="optional"/>
1588 </complexType>
1589 <element name="Extensions" type="sampl:ExtensionsType"/>
1590 <complexType name="ExtensionsType">
1591   <sequence>
1592     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1593   </sequence>
1594 </complexType>
```

1595 3.2.2 Complex Type StatusResponseType

1596 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
1597 defines common attributes and elements that are associated with all SAML responses:

1598 ID [Required]

1599 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in
1600 Section 1.3.4 for identifier uniqueness.

1601 InResponseTo [Optional]

1602 A reference to the identifier of the request to which the response corresponds, if any. If the response
1603 is not generated in response to a request, or if the ID attribute value of a request cannot be
1604 determined (for example, the request is malformed), then this attribute MUST NOT be present.
1605 Otherwise, it MUST be present and its value MUST match the value of the corresponding request's ID
1606 attribute.

1607 Version [Required]

1608 The version of this response. The identifier for the version of SAML defined in this specification is
1609 "2.0". SAML versioning is discussed in Section 4.

1610 IssueInstant [Required]

1611 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
1612 1.3.3.

1613 Destination [Optional]

1614 A URI reference indicating the address to which this response has been sent. This is useful to prevent
1615 malicious forwarding of responses to unintended recipients, a protection that is required by some
1616 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the
1617 location at which the message was received. If it does not, the response MUST be discarded. Some
1618 protocol bindings may require the use of this attribute (see [SAMLBind]).

1619 Consent [Optional]

1620 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1621 the sending of this response. See Section 8.4 for some URI references that MAY be used as the value
1622 of the Consent attribute and their associated descriptions. If no Consent value is provided, the
1623 identifier urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in
1624 effect.

1625 <saml:Issuer> [Optional]

1626 Identifies the entity that generated the response message. (For more information on this element, see
1627 Section 2.2.5.)

1628 <ds:Signature> [Optional]

1629 An XML Signature that authenticates the responder and provides message integrity, as described
1630 below and in Section 5.

1631 <Extensions> [Optional]

1632 This extension point contains optional protocol message extension elements that are agreed on
1633 between the communicating parties. . No extension schema is required in order to make use of this
1634 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1635 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1636 SAML-defined namespace.

1637 <Status> [Required]

1638 A code representing the status of the corresponding request.

1639 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to
1640 authenticate itself, and message integrity may often be required. Authentication and message integrity
1641 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML
1642 response MAY be signed, which provides both authentication of the responder and message integrity.

1643 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML
1644 requester receiving the response MUST verify that the signature is valid (that is, that the message has not
1645 been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on
1646 the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD
1647 evaluate the signature to determine the identity and appropriateness of the signer and may continue to
1648 process the response as it deems appropriate.

1649 If a `Consent` attribute is included and the value indicates that some form of principal consent has been
1650 obtained, then the response SHOULD be signed.

1651 The following schema fragment defines the **StatusResponseType** complex type:

```
1652 <complexType name="StatusResponseType">  
1653   <sequence>  
1654     <element ref="saml:Issuer" minOccurs="0"/>  
1655     <element ref="ds:Signature" minOccurs="0"/>  
1656     <element ref="samlp:Extensions" minOccurs="0"/>  
1657     <element ref="samlp:Status"/>  
1658   </sequence>  
1659   <attribute name="ID" type="ID" use="required"/>  
1660   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1661   <attribute name="Version" type="string" use="required"/>  
1662   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1663   <attribute name="Destination" type="anyURI" use="optional"/>  
1664   <attribute name="Consent" type="anyURI" use="optional"/>  
1665 </complexType>
```

1666 3.2.2.1 Element `<Status>`

1667 The `<Status>` element contains the following elements:

1668 `<StatusCode>` [Required]

1669 A code representing the status of the activity carried out in response to the corresponding request.

1670 `<StatusMessage>` [Optional]

1671 A message which MAY be returned to an operator.

1672 `<StatusDetail>` [Optional]

1673 Additional information concerning the status of the request.

1674 The following schema fragment defines the `<Status>` element and its **StatusType** complex type:

```
1675 <element name="Status" type="samlp:StatusType"/>  
1676 <complexType name="StatusType">  
1677   <sequence>  
1678     <element ref="samlp:StatusCode"/>  
1679     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1680     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1681   </sequence>  
1682 </complexType>
```

1683 3.2.2.2 Element `<StatusCode>`

1684 The `<StatusCode>` element specifies a code or a set of nested codes representing the status of the
1685 corresponding request. The `<StatusCode>` element has the following element and attribute:

1686 Value [Required]

1687 The status code value. This attribute contains a URI reference. The value of the topmost
1688 <StatusCode> element MUST be from the top-level list provided in this section.

1689 <StatusCode> [Optional]

1690 A subordinate status code that provides more specific information on an error condition. Note that
1691 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for
1692 additional information by intentionally presenting erroneous requests.

1693 The permissible top-level <StatusCode> values are as follows:

1694 urn:oasis:names:tc:SAML:2.0:status:Success

1695 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1696 <StatusDetail> elements.

1697 urn:oasis:names:tc:SAML:2.0:status:Requester

1698 The request could not be performed due to an error on the part of the requester.

1699 urn:oasis:names:tc:SAML:2.0:status:Responder

1700 The request could not be performed due to an error on the part of the SAML responder or SAML
1701 authority.

1702 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1703 The SAML responder could not process the request because the version of the request message was
1704 incorrect.

1705 The following second-level status codes are referenced at various places in this specification. Additional
1706 second-level status codes MAY be defined in future versions of the SAML specification. System entities
1707 are free to define more specific status codes by defining appropriate URI references.

1708 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1709 The responding provider was unable to successfully authenticate the principal.

1710 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1711 Unexpected or invalid content was encountered within a <saml:Attribute> or
1712 <saml:AttributeValue> element.

1713 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1714 The responding provider cannot or will not support the requested name identifier policy.

1715 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1716 The specified authentication context requirements cannot be met by the responder.

1717 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1718 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an
1719 <IDPList> can be resolved or that none of the supported identity providers are available.

1720 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1721 Indicates the responding provider cannot authenticate the principal passively, as has been requested.

1722 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP

1723 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1724 supported by the intermediary.

1725 urn:oasis:names:tc:SAML:2.0:status:PartialLogout
 1726 Used by a session authority to indicate to a session participant that it was not able to propagate logout
 1727 to all other session participants.

1728 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
 1729 Indicates that a responding provider cannot authenticate the principal directly and is not permitted to
 1730 proxy the request further.

1731 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
 1732 The SAML responder or SAML authority is able to process the request but has chosen not to respond.
 1733 This status code MAY be used when there is concern about the security context of the request
 1734 message or the sequence of request messages received from a particular requester.

1735 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
 1736 The SAML responder or SAML authority does not support the request.

1737 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
 1738 The SAML responder cannot process any requests with the protocol version specified in the request.

1739 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
 1740 The SAML responder cannot process the request because the protocol version specified in the
 1741 request message is a major upgrade from the highest protocol version supported by the responder.

1742 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
 1743 The SAML responder cannot process the request because the protocol version specified in the
 1744 request message is too low.

1745 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
 1746 The resource value provided in the request message is invalid or unrecognized.

1747 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
 1748 The response message would contain more elements than the SAML responder is able to return.

1749 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
 1750 An entity that has no knowledge of a particular attribute profile has been presented with an attribute
 1751 drawn from that profile.

1752 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
 1753 The responding provider does not recognize the principal specified or implied by the request.

1754 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
 1755 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
 1756 request.

1757 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
 1758 type:

```

1759 <element name="StatusCode" type="samlp:StatusCodeType"/>
1760 <complexType name="StatusCodeType">
1761   <sequence>
1762     <element ref="samlp:StatusCode" minOccurs="0"/>
1763   </sequence>
1764   <attribute name="Value" type="anyURI" use="required"/>
1765 </complexType>

```

1766 3.2.2.3 Element <StatusMessage>

1767 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1768 The following schema fragment defines the <StatusMessage> element:

```
1769 <element name="StatusMessage" type="string"/>
```

1770 3.2.2.4 Element <StatusDetail>

1771 The <StatusDetail> element MAY be used to specify additional information concerning the status of
1772 the request. The additional information consists of zero or more elements from any namespace, with no
1773 requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

1774 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1775 complex type:

```
1776 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1777 <complexType name="StatusDetailType">  
1778   <sequence>  
1779     <any namespace="##any" processContents="lax" minOccurs="0"  
1780     maxOccurs="unbounded"/>  
1781   </sequence>  
1782 </complexType>
```

1783 3.3 Assertion Query and Request Protocol

1784 This section defines messages and processing rules for requesting existing assertions by reference or
1785 querying for assertions by subject and statement type.

1786 3.3.1 Element <AssertionIDRequest>

1787 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>
1788 message element can be used to request that they be returned in a <Response> message. The
1789 <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for
1790 more information on this element.

1791 The following schema fragment defines the <AssertionIDRequest> element:

```
1792 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>  
1793 <complexType name="AssertionIDRequestType">  
1794   <complexContent>  
1795     <extension base="samlp:RequestAbstractType">  
1796       <sequence>  
1797         <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>  
1798       </sequence>  
1799     </extension>  
1800   </complexContent>  
1801 </complexType>
```

1802 3.3.2 Queries

1803 The following sections define the SAML query request messages.

1804 3.3.2.1 Element <SubjectQuery>

1805 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1806 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and

1807 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the
1808 <saml:Subject> element (defined in Section 2.4) to **RequestAbstractType**.

1809 The following schema fragment defines the <SubjectQuery> element and its
1810 **SubjectQueryAbstractType** complex type:

```
1811 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1812 <complexType name="SubjectQueryAbstractType" abstract="true">  
1813   <complexContent>  
1814     <extension base="samlp:RequestAbstractType">  
1815       <sequence>  
1816         <element ref="saml:Subject"/>  
1817       </sequence>  
1818     </extension>  
1819   </complexContent>  
1820 </complexType>
```

1821 3.3.2.2 Element <AuthnQuery>

1822 The <AuthnQuery> message element is used to make the query “What assertions containing
1823 authentication statements are available for this subject?” A successful <Response> will contain one or
1824 more assertions containing authentication statements.

1825 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1826 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1827 that have occurred in a previous interaction between the indicated subject and the authentication authority.

1828 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1829 the following element and attribute:

1830 SessionIndex [Optional]

1831 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1832 containing authentication statements do you have for this subject within the context of the supplied
1833 session information?”

1834 <RequestedAuthnContext> [Optional]

1835 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1836 containing authentication statements do you have for this subject that satisfy the authentication
1837 context requirements in this element?”

1838 In response to an authentication query, a SAML authority returns assertions with authentication
1839 statements as follows:

- 1840 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1841 assertions that may be returned.
- 1842 • If the SessionIndex attribute is present in the query, at least one <AuthnStatement> element in
1843 the set of returned assertions MUST contain a SessionIndex attribute that matches the
1844 SessionIndex attribute in the query. It is OPTIONAL for the complete set of all such matching
1845 assertions to be returned in the response.
- 1846 • If the <RequestedAuthnContext> element is present in the query, at least one
1847 <AuthnStatement> element in the set of returned assertions MUST contain an
1848 <AuthnContext> element that satisfies the element in the query (see Section 3.3.2.2.1). It is
1849 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1850 The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex
1851 type:

```
1852 <element name="AuthnQuery" type="samlp:AuthnQueryType"/>  
1853 <complexType name="AuthnQueryType">
```

```

1854     <complexContent>
1855         <extension base="saml:SubjectQueryAbstractType">
1856             <sequence>
1857                 <element ref="saml:RequestedAuthnContext" minOccurs="0"/>
1858             </sequence>
1859             <attribute name="SessionIndex" type="string" use="optional"/>
1860         </extension>
1861     </complexContent>
1862 </complexType>

```

1863 3.3.2.2.1 Element <RequestedAuthnContext>

1864 The <RequestedAuthnContext> element specifies the authentication context requirements of
 1865 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
 1866 complex type defines the following elements and attributes:

1867 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1868 Specifies one or more URI references identifying authentication context classes or declarations.
 1869 These elements are defined in Section 2.7.2.2. For more information about authentication context
 1870 classes, see [SAMLAuthnCxt].

1871 Comparison [Optional]

1872 Specifies the comparison method used to evaluate the requested context classes or statements, one
 1873 of "exact", "minimum", "maximum", or "better". The default is "exact".

1874 Either a set of class references or a set of declaration references can be used. [\[E45\]If ordering is relevant](#)
 1875 [to the evaluation of the request, then the](#) set of supplied references MUST be evaluated as an ordered
 1876 set, where the first element is the most preferred authentication context class or declaration. [For example,](#)
 1877 [ordering is significant when using this element in an <AuthnRequest> message but not in an](#)
 1878 [<AuthnQuery> message.](#)

1879 If none of the specified classes or declarations can be satisfied in accordance with the rules below, then
 1880 the responder MUST return a <Response> message with a [\[E65\]top-level <StatusCode> of](#)
 1881 [urn:oasis:names:tc:SAML:2.0:status:Responder](#) and MAY return a second-level
 1882 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1883 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
 1884 statement MUST be the exact match of at least one of the authentication contexts specified.

1885 If Comparison is set to "minimum", then the resulting authentication context in the authentication
 1886 statement MUST be at least as strong (as deemed by the responder) as one of the authentication
 1887 contexts specified.

1888 If Comparison is set to "better", then the resulting authentication context in the authentication
 1889 statement MUST be stronger (as deemed by the responder) than [\[E45\]any](#) one of the authentication
 1890 contexts specified.

1891 If Comparison is set to "maximum", then the resulting authentication context in the authentication
 1892 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength
 1893 of at least one of the authentication contexts specified.

1894 The following schema fragment defines the <RequestedAuthnContext> element and its
 1895 **RequestedAuthnContextType** complex type:

```

1896     <element name="RequestedAuthnContext" type="saml:RequestedAuthnContextType"/>
1897     <complexType name="RequestedAuthnContextType">
1898         <choice>
1899             <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1900             <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>

```

```

1901     </choice>
1902     <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1903 use="optional"/>
1904 </complexType>
1905 <simpleType name="AuthnContextComparisonType">
1906     <restriction base="string">
1907         <enumeration value="exact"/>
1908         <enumeration value="minimum"/>
1909         <enumeration value="maximum"/>
1910         <enumeration value="better"/>
1911     </restriction>
1912 </simpleType>

```

1913 3.3.2.3 Element <AttributeQuery>

1914 The <AttributeQuery> element is used to make the query “Return the requested attributes for this
1915 subject.” A successful response will be in the form of assertions containing attribute statements, to the
1916 extent allowed by policy. This element is of type **AttributeQueryType**, which extends
1917 **SubjectQueryAbstractType** with the addition of the following element:

1918 <saml:Attribute> [Any Number]

1919 Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no
1920 attributes are specified, it indicates that all attributes allowed by policy are requested. If a given
1921 <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if
1922 that attribute is returned in the response, it MUST NOT contain any values that are not equal to the
1923 values specified in the query. In the absence of equality rules specified by particular profiles or
1924 attributes, equality is defined as an identical XML representation of the value. For more information on
1925 <saml:Attribute>, see Section 2.7.3.1.

1926 A single query MUST NOT contain two <saml:Attribute> elements with the same Name and
1927 NameFormat values (that is, a given attribute MUST be named only once in a query).

1928 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1929 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1930 assertions that may be returned.
- 1931 • If any <Attribute> elements are present in the query, they constrain/filter the attributes and
1932 optionally the values returned, as noted above.
- 1933 • The attributes and values returned MAY also be constrained by application-specific policy
1934 considerations.

1935 The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
1936 and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to
1937 indicate problems with the interpretation of attribute or value information in a query.

1938 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1939 complex type:

```

1940 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1941 <complexType name="AttributeQueryType">
1942     <complexContent>
1943         <extension base="samlp:SubjectQueryAbstractType">
1944             <sequence>
1945                 <element ref="saml:Attribute" minOccurs="0"
1946 maxOccurs="unbounded"/>
1947             </sequence>
1948         </extension>
1949     </complexContent>
1950 </complexType>

```

1951 **3.3.2.4 Element <AuthzDecisionQuery>**

1952 The <AuthzDecisionQuery> element is used to make the query "Should these actions on this resource
1953 be allowed for this subject, given this evidence?" A successful response will be in the form of assertions
1954 containing authorization decision statements.

1955 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no
1956 future enhancements planned. Users who require additional functionality may want to
1957 consider the eXtensible Access Control Markup Language [XACML], which offers
1958 enhanced authorization decision features.

1959 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1960 addition of the following elements and attribute:

1961 Resource [Required]

1962 A URI reference indicating the resource for which authorization is requested.

1963 <saml:Action> [One or More]

1964 The actions for which authorization is requested. For more information on this element, see Section
1965 2.7.4.2.

1966 <saml:Evidence> [Optional]

1967 A set of assertions that the SAML authority MAY rely on in making its authorization decision. For more
1968 information on this element, see Section 2.7.4.3.

1969 In response to an authorization decision query, a SAML authority returns assertions with authorization
1970 decision statements as follows:

- 1971 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1972 assertions that may be returned.

1973 The following schema fragment defines the <AuthzDecisionQuery> element and its
1974 **AuthzDecisionQueryType** complex type:

```
1975 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>  
1976 <complexType name="AuthzDecisionQueryType">  
1977   <complexContent>  
1978     <extension base="samlp:SubjectQueryAbstractType">  
1979       <sequence>  
1980         <element ref="saml:Action" maxOccurs="unbounded"/>  
1981         <element ref="saml:Evidence" minOccurs="0"/>  
1982       </sequence>  
1983       <attribute name="Resource" type="anyURI" use="required"/>  
1984     </extension>  
1985   </complexContent>  
1986 </complexType>
```

1987 **3.3.3 Element <Response>**

1988 The <Response> message element is used when a response consists of a list of zero or more assertions
1989 that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType**
1990 and adds the following elements:

1991 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1992 Specifies an assertion by value, or optionally an encrypted assertion by value. See Section 2.3.3 for
1993 more information on these elements.

1994 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1995 <element name="Response" type="samlp:ResponseType"/>
1996 <complexType name="ResponseType">
1997   <complexContent>
1998     <extension base="samlp:StatusResponseType">
1999       <choice minOccurs="0" maxOccurs="unbounded">
2000         <element ref="saml:Assertion"/>
2001         <element ref="saml:EncryptedAssertion"/>
2002       </choice>
2003     </extension>
2004   </complexContent>
2005 </complexType>

```

2006 3.3.4 Processing Rules

2007 In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST**
 2008 contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the
 2009 query.

2010 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both
 2011 apply:

- 2012 • If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST**
 2013 include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2.
 2014 In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical"
 2015 means that the identifier element's content and attribute values **MUST** be the same. An encrypted
 2016 identifier will be identical to the original according to this definition, once decrypted.
- 2017 • If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at
 2018 least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner
 2019 described by at least one `<saml:SubjectConfirmation>` element in S2.

2020 As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular
 2021 `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting
 2022 S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with
 2023 different `Format` values and element content, even if the two identifiers are considered to refer to the
 2024 same principal.

2025 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
 2026 expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an
 2027 `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value
 2028 `urn:oasis:names:tc:SAML:2.0:status:Success`.

2029 All other processing rules associated with the underlying request and response messages **MUST** be
 2030 observed.

2031 3.4 Authentication Request Protocol

2032 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
 2033 authentication statements to establish a security context at one or more relying parties, it can use the
 2034 authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and
 2035 request that it return a `<Response>` message containing one or more such assertions. Such assertions
 2036 **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one
 2037 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

2038 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
 2039 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
 2040 is not specified, though the means of authentication might impact the content of the response. Other
 2041 issues related to the validation of authentication credentials by the identity provider or any communication

2042 between the identity provider and any other entities involved in the authentication process are also out of
2043 scope of this protocol.

2044 The descriptions and processing rules in the following sections reference the following actors, many of
2045 whom might be the same entity in a particular profile of use:

2046 **Requester**

2047 The entity who creates the authentication request and to whom the response is to be returned.

2048 **Presenter**

2049 The entity who presents the request to the identity provider and either authenticates itself during
2050 the transmission of the message, or relies on an existing security context to establish its identity. If
2051 not the requester, the presenter acts as an intermediary between the requester and the
2052 responding identity provider.

2053 **Requested Subject**

2054 The entity about whom one or more assertions are being requested.

2055 **Attesting Entity**

2056 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`
2057 elements of the resulting assertion(s).

2058 **Relying Party**

2059 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
2060 the profile or context of use, generally to establish a security context.

2061 **Identity Provider**

2062 The entity to whom the presenter gives the request and from whom the presenter receives the
2063 response.

2064 **3.4.1 Element `<AuthnRequest>`**

2065 To request that an identity provider issue an assertion with an authentication statement, a presenter
2066 authenticates to that identity provider (or relies on an existing security context) and sends it an
2067 `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to
2068 satisfy its purpose. Among these properties may be information that relates to the content of the assertion
2069 and/or information that relates to how the resulting `<Response>` message should be delivered to the
2070 requester. The process of authentication of the presenter may take place before, during, or after the initial
2071 delivery of the `<AuthnRequest>` message.

2072 The requester might not be the same as the presenter of the request if, for example, the requester is a
2073 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject
2074 so that the relying party can decide whether to provide a service.

2075 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2076 by the protocol binding used to deliver the message.

2077 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
2078 adds the following elements and attributes, all of which are optional in general, but may be required by
2079 specific profiles:

2080 `<saml:Subject>` [Optional]

2081 Specifies the requested subject of the resulting assertion(s). This may include one or more
2082 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions
2083 can be confirmed. For more information on this element, see Section 2.4.

2084 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
2085 requested subject. If no `<saml:SubjectConfirmation>` elements are included, then the presenter
2086 is presumed to be the only attesting entity required and the method is implied by the profile of use
2087 and/or the policies of the identity provider.

2088 `<NameIDPolicy>` [Optional]

2089 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
2090 then any type of identifier supported by the identity provider for the requested subject can be used,
2091 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2092 `<saml:Conditions>` [Optional]

2093 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting
2094 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
2095 information in this element is used as input to the process of constructing the assertion, rather than as
2096 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2097 `<RequestedAuthnContext>` [Optional]

2098 Specifies the requirements, if any, that the requester places on the authentication context that applies
2099 to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for processing rules
2100 regarding this element.

2101 `<Scoping>` [Optional]

2102 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as
2103 limitations and context related to proxying of the `<AuthnRequest>` message to subsequent identity
2104 providers by the responder.

2105 `ForceAuthn` [Optional]

2106 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
2107 rely on a previous security context. If a value is not provided, the default is "false". However, if both
2108 `ForceAuthn` and `IsPassive` are "true", the identity provider MUST NOT freshly authenticate the
2109 presenter unless the constraints of `IsPassive` can be met.

2110 `IsPassive` [Optional]

2111 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take control
2112 of the user interface from the requester and interact with the presenter in a noticeable fashion. If a
2113 value is not provided, the default is "false".

2114 `AssertionConsumerServiceIndex` [Optional]

2115 Indirectly identifies the location to which the `<Response>` message should be returned to the
2116 requester. It applies only to profiles in which the requester is different from the presenter, such as the
2117 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map
2118 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one
2119 possible mechanism. If omitted, then the identity provider MUST return the `<Response>` message to
2120 the default location associated with the requester for the profile of use. If the index specified is invalid,
2121 then the identity provider MAY return an error `<Response>` or it MAY use the default location. This
2122 attribute is mutually exclusive with the `AssertionConsumerServiceURL` and `ProtocolBinding`
2123 attributes.

2124 `AssertionConsumerServiceURL` [Optional]

2125 Specifies by value the location to which the `<Response>` message MUST be returned to the
2126 requester. The responder MUST ensure by some means that the value specified is in fact associated
2127 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing
2128 `<AuthnRequest>` message is another. This attribute is mutually exclusive with the
2129 `AssertionConsumerServiceIndex` attribute and is typically accompanied by the
2130 `ProtocolBinding` attribute.

2131 ProtocolBinding [Optional]

2132 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>
2133 message. See [SAMLBind] for more information about protocol bindings and URI references defined
2134 for them. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute
2135 and is typically accompanied by the AssertionConsumerServiceURL attribute.

2136 AttributeConsumingServiceIndex [Optional]

2137 Indirectly identifies information associated with the requester describing the SAML attributes the
2138 requester desires or requires to be supplied by the identity provider in the <Response> message. The
2139 identity provider MUST have a trusted means to map the index value in the attribute to information
2140 associated with the requester. [SAMLMeta] provides one possible mechanism. The identity provider
2141 MAY use this information to populate one or more <saml:AttributeStatement> elements in the
2142 assertion(s) it returns.

2143 ProviderName [Optional]

2144 Specifies the human-readable name of the requester for use by the presenter's user agent or the
2145 identity provider.

2146 See Section 3.4.1.4 for general processing rules regarding this message.

2147 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**
2148 complex type:

```
2149 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
2150 <complexType name="AuthnRequestType">
2151   <complexContent>
2152     <extension base="samlp:RequestAbstractType">
2153       <sequence>
2154         <element ref="saml:Subject" minOccurs="0"/>
2155         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
2156         <element ref="saml:Conditions" minOccurs="0"/>
2157         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
2158         <element ref="samlp:Scoping" minOccurs="0"/>
2159       </sequence>
2160       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2161       <attribute name="IsPassive" type="boolean" use="optional"/>
2162       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2163       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2164 use="optional"/>
2165       <attribute name="AssertionConsumerServiceURL" type="anyURI"
2166 use="optional"/>
2167       <attribute name="AttributeConsumingServiceIndex"
2168 type="unsignedShort" use="optional"/>
2169       <attribute name="ProviderName" type="string" use="optional"/>
2170     </extension>
2171   </complexContent>
2172 </complexType>
```

2173 3.4.1.1 Element <NameIDPolicy>

2174 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
2175 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

2176 Format [Optional]

2177 Specifies the URI reference corresponding to a name identifier format defined in this or another
2178 specification (see Section 8.3 for examples). The additional value of
2179 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use
2180 within this attribute to indicate a request that the resulting identifier be encrypted.

2181 SPNameQualifier [Optional]

2182 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of
2183 a service provider other than the requester, or in the namespace of an affiliation group of service
2184 providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-`
2185 `format:persistent` in Section 8.3.7.

2186 AllowCreate [Optional]

2187 A Boolean value used to indicate whether [\[E 14\]the requester grants to](#) the identity provider ~~is allowed,~~
2188 in the course of fulfilling the request, [permission](#) to create a new identifier ~~to represent the principal or~~
2189 [to associate an existing identifier representing the principal with the relying party](#). Defaults to "false":
2190 ~~When "false", the requester constrains the identity provider to only issue an assertion to it if an~~
2191 ~~acceptable identifier for the principal has already been established. Note that this does not prevent the~~
2192 ~~identity provider from creating such identifiers outside the context of this specific request (for example,~~
2193 ~~in advance for a large number of principals)-if not present or the entire element is omitted.~~

2194 [The AllowCreate attribute may be used by some deployments to influence the creation of state](#)
2195 [maintained by the identity provider pertaining to the use of a name identifier \(or any other persistent,](#)
2196 [uniquely identifying attributes\) by a particular relying party, for purposes such as dynamic identifier or](#)
2197 [attribute creation, tracking of consent, subsequent use of the Name Identifier Management protocol](#)
2198 [\(see Section 3.6\), or other related purposes.](#)

2199 [When "false", the requester tries to constrain the identity provider to issue an assertion only if such](#)
2200 [state has already been established or is not deemed applicable by the identity provider to the use of](#)
2201 [an identifier. Thus, this does not prevent the identity provider from assuming such information exists](#)
2202 [outside the context of this specific request \(for example, establishing it in advance for a large number](#)
2203 [of principals\).](#)

2204 [A value of "true" permits the identity provider to take any related actions it wishes to fulfill the request,](#)
2205 [subject to any other constraints imposed by the request and policy \(the `IsPassive` attribute, for](#)
2206 [example\).](#)

2207 [Generally, requesters cannot assume specific behavior from identity providers regarding the initial](#)
2208 [creation or association of identifiers on their behalf, as these are details left to implementations or](#)
2209 [deployments. Absent specific profiles governing the use of this attribute, it might be used as a hint to](#)
2210 [identity providers about the requester's intention to store the identifier or link it to a local value.](#)

2211 [A value of "false" might be used to indicate that the requester is not prepared or able to do so and](#)
2212 [save the identity provider wasted effort.](#)

2213 [Requesters that do not make specific use of this attribute SHOULD generally set it to "true" to](#)
2214 [maximize interoperability.](#)

2215 [The use of the AllowCreate attribute MUST NOT be used and SHOULD be ignored in conjunction with](#)
2216 [requests for or assertions issued with name identifiers with a `Format` of](#)
2217 [`urn:oasis:names:tc:SAML:2.0:nameid-format:transient` \(they preclude any such state in](#)
2218 [and of themselves\).](#)

2219 When this element is used, if the content is not understood by or acceptable to the identity provider, then a
2220 <Response> message element MUST be returned with an error <Status>, and MAY contain a second-
2221 level <StatusCode> of `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

2222 If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-`
2223 `format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any
2224 additional constraints due to the content of this element or the policies of the identity provider or principal.

2225 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates
2226 that the resulting assertion(s) MUST contain <EncryptedID> elements instead of plaintext. The
2227 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the
2228 requested subject. [\[E6\]It is not possible for the service provider to specifically request that a particular kind](#)

2229 of identifier be returned if it asks for encryption. The `<md:NameIDFormat>` metadata element (see
2230 [SAMLMeta]) or other out-of-band means MAY be used to determine what kind of identifier to encrypt and
2231 return.

2232 [E15]When a Format defined in Section 8.3 other than `urn:oasis:names:tc:SAML:1.1:nameid-`
2233 `format:unspecified` Or `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` is used,
2234 then if the identity provider returns any assertions:

- 2235 • the Format value of the `<NameID>` within the `<Subject>` of any `<Assertion>` MUST be identical
2236 to the Format value supplied in the `<NameIDPolicy>`, and
- 2237 • if `SPNameQualifier` is not omitted in `<NameIDPolicy>`, the `SPNameQualifier` value of the
2238 `<NameID>` within the `<Subject>` of any `<Assertion>` MUST be identical to the
2239 `SPNameQualifier` value supplied in the `<NameIDPolicy>`.

2240 Regardless of the Format in the `<NameIDPolicy>`, the identity provider MAY return an
2241 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly
2242 specific to the service provider) require that an encrypted identifier be used.

2243 [E14]Note that if the requester wishes to permit the identity provider to establish a new identifier for the
2244 principal if none exists, it MUST include this element with the `AllowCreate` attribute set to "true".
2245 Otherwise, only a principal for whom the identity provider has previously established an identifier usable by
2246 the requester can be authenticated successfully. This is primarily useful in conjunction with the
2247 `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` Format value (see Section 8.3.7).

2248 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
2249 complex type:

```
2250 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>  
2251 <complexType name="NameIDPolicyType">  
2252   <attribute name="Format" type="anyURI" use="optional"/>  
2253   <attribute name="SPNameQualifier" type="string" use="optional"/>  
2254   <attribute name="AllowCreate" type="boolean" use="optional"/>  
2255 </complexType>
```

2256 3.4.1.2 Element `<Scoping>`

2257 The `<Scoping>` element specifies the identity providers trusted by the requester to authenticate the
2258 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
2259 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
2260 elements and attribute:

2261 `ProxyCount` [Optional]

2262 Specifies the number of proxying indirections permissible between the identity provider that receives
2263 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of
2264 zero permits no proxying, while omitting this attribute expresses no such restriction.

2265 `<IDPList>` [Optional]

2266 An advisory list of identity providers and associated information that the requester deems acceptable
2267 to respond to the request.

2268 `<RequesterID>` [Zero or More]

2269 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate
2270 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for a
2271 description of entity identifiers.

2272 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
2273 <Response> message with an error <Status> and [E65]optionally a second-level <StatusCode> of
2274 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP Or
2275 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support
2276 any of the specified identity providers.

2277 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```
2278 <element name="Scoping" type="samlp:ScopingType"/>
2279 <complexType name="ScopingType">
2280   <sequence>
2281     <element ref="samlp:IDPList" minOccurs="0"/>
2282     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2283   </sequence>
2284   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2285 </complexType>
2286 <element name="RequesterID" type="anyURI"/>
```

2287 3.4.1.3 Element <IDPList>

2288 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the
2289 presenter. Its **IDPListType** complex type defines the following elements:

2290 <IDPEntry> [One or More]

2291 Information about a single identity provider.

2292 <GetComplete> [Optional]

2293 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to
2294 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML
2295 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>
2296 element.

2297 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
2298 <element name="IDPList" type="samlp:IDPListType"/>
2299 <complexType name="IDPListType">
2300   <sequence>
2301     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2302     <element ref="samlp:GetComplete" minOccurs="0"/>
2303   </sequence>
2304 </complexType>
2305 <element name="GetComplete" type="anyURI"/>
```

2306 3.4.1.3.1 Element <IDPEntry>

2307 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the
2308 presenter. Its **IDPEntryType** complex type defines the following attributes:

2309 ProviderID [Required]

2310 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2311 Name [Optional]

2312 A human-readable name for the identity provider.

2313 Loc [Optional]

2314 A URI reference representing the location of a profile-specific endpoint supporting the authentication
2315 request protocol. The binding to be used must be understood from the profile of use.

2316 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```

2317 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2318 <complexType name="IDPEntryType">
2319   <attribute name="ProviderID" type="anyURI" use="required"/>
2320   <attribute name="Name" type="string" use="optional"/>
2321   <attribute name="Loc" type="anyURI" use="optional"/>
2322 </complexType>

```

2323 3.4.1.4 Processing Rules

2324 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
 2325 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
 2326 kinds of input and output are required or prohibited. The following processing rules apply as invariant
 2327 behavior across any profile of this protocol exchange. All other processing rules associated with the
 2328 underlying request and response messages MUST also be observed.

2329 The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing
 2330 one or more assertions that meet the specifications defined by the request, or with a <Response>
 2331 message containing a <Status> describing the error that occurred. The responder MAY conduct
 2332 additional message exchanges with the presenter as needed to initiate or complete the authentication
 2333 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
 2334 the next section, this includes proxying the request by directing the presenter to another identity provider
 2335 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to
 2336 authenticate the presenter to the original responder, in effect using SAML as the authentication
 2337 mechanism.

2338 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if
 2339 prevented from providing an assertion by policies in effect at the identity provider (for example the
 2340 intended subject has prohibited the identity provider from providing assertions to the relying party), then it
 2341 MUST return a <Response> with an error <Status>, and MAY return a second-level <StatusCode> of
 2342 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
 2343 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2344 If the <saml:Subject> element in the request is present, then the resulting assertions'
 2345 <saml:Subject> MUST **strongly match** the request <saml:Subject>, as described in Section 3.3.4,
 2346 except that the identifier MAY be in a different format if specified by <NameIDPolicy>. In such a case,
 2347 the identifier's physical content MAY be different, but it MUST refer to the same principal. [E75]If the
 2348 identity provider cannot or will not produce assertions with a strongly matching subject, then it MUST
 2349 return a <Response> with an error <Status>, and MAY return a second-level <StatusCode> that
 2350 reflects the reason for the failure.

2351 All of the content defined specifically within <AuthnRequest> is optional, although some may be required
 2352 by certain profiles. In the absence of any specific content at all, the following behavior is implied:

- 2353 • The assertion(s) returned MUST contain a <saml:Subject> element that represents the
 2354 presenter. The identifier type and format are determined by the identity provider. At least one
 2355 statement in at least one assertion MUST be a <saml:AuthnStatement> that describes the
 2356 authentication performed by the responder or authentication service associated with it.
- 2357 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the
 2358 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation
 2359 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- 2360 • The resulting assertion(s) MUST contain a <saml:AudienceRestriction> element
 2361 referencing the requester as an acceptable relying party. Other audiences MAY be included as
 2362 deemed appropriate by the identity provider.

2363 3.4.1.5 Proxying

2364 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or
2365 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
2366 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
2367 <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in
2368 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
2369 identity provider.

2370 Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the
2371 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
2372 proxying provider can issue an assertion of its own in response to the original <AuthnRequest>,
2373 completing the overall message exchange. Both the proxying and authenticating identity providers MAY
2374 include constraints on proxying activity in the messages and assertions they issue, as described in
2375 previous sections and below.

2376 The requester can influence proxy behavior by including a <Scoping> element where the provider sets a
2377 desired ProxyCount value and/or indicates a list of preferred identity providers which may be proxied by
2378 including an ordered <IDPList> of preferred providers.

2379 An identity provider can control secondary use of its assertions by proxying identity providers using a
2380 <ProxyRestriction> element in the assertions it issues.

2381 3.4.1.5.1 Proxying Processing Rules

2382 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is
2383 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2384 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

2385 An identity provider MUST NOT proxy a request where <ProxyCount> is set to zero. [65]Unless the
2386 identity provider can directly authenticate the presenter, ifThe identity provider MUST return a
2387 <Response> message with an error top level <StatusCode> value of
2388 urn:oasis:names:tc:SAML:2.0:status:Responder containing and may return a second-level
2389 <StatusCode> value of urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded,
2390 unless it can directly authenticate the presenter.

2391 If it chooses to proxy to a SAML identity provider, when creating the new <AuthnRequest>, the proxying
2392 identity provider MUST include equivalent or stricter forms of all the information included in the original
2393 request (such as authentication context policy). Note, however, that the proxying provider is free to specify
2394 whatever <NameIDPolicy> it wishes to maximize the chances of a successful response.

2395 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
2396 have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for
2397 example) will be honored by the authenticating provider.

2398 The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less
2399 than the original value. If the original request does not contain a <ProxyCount> attribute, then the new
2400 request SHOULD contain a <ProxyCount> attribute.

2401 If an <IDPList> was specified in the original request, the new request MUST also contain an
2402 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the
2403 <IDPList>, but MUST NOT remove any from the list.

2404 The authentication request and response are processed in normal fashion, in accordance with the rules
2405 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
2406 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2407 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
2408 relevant information from the original assertion or non-SAML equivalent.
- 2409 • The new assertion's `<saml:Subject>` MUST contain an identifier that satisfies the original
2410 requester's preferences, as defined by its `<NameIDPolicy>` element.
- 2411 • The `<saml:AuthnStatement>` in the new assertion MUST include a `<saml:AuthnContext>`
2412 element containing a `<saml:AuthenticatingAuthority>` element referencing the identity
2413 provider to which the proxying identity provider referred the presenter. If the original assertion
2414 contains `<saml:AuthnContext>` information that includes one or more
2415 `<saml:AuthenticatingAuthority>` elements, those elements SHOULD be included in the
2416 new assertion, with the new element placed after them.
- 2417 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
2418 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
2419 consistent over time across different requests. The value MUST not conflict with values used or
2420 generated by other SAML providers.
- 2421 • Any other `<saml:AuthnContext>` information MAY be copied, translated, or omitted in
2422 accordance with the policies of the proxying identity provider, provided that the original
2423 requirements dictated by the requester are met.

2424 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,
2425 and this request is equally or less strict than the original request (as determined by the proxying identity
2426 provider), the identity provider MAY skip the creation of a new `<AuthnRequest>` to the authenticating
2427 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML
2428 equivalent it received is still valid).

2429 **3.5 Artifact Resolution Protocol**

2430 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
2431 transported in a SAML binding by reference instead of by value. Both requests and responses can be
2432 obtained by reference using this specialized protocol. A message sender, instead of binding a message to
2433 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2434 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver
2435 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
2436 protocol to resolve the artifact into the original protocol message.

2437 The most common use for this mechanism is with bindings that cannot easily carry a message because of
2438 size constraints, or to enable a message to be communicated via a secure channel between the SAML
2439 requester and responder, avoiding the need for a signature.

2440 Depending on the characteristics of the underlying message being passed by reference, the artifact
2441 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2442 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
2443 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
2444 by any party.

2445 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
2446 as if the message so obtained had been sent originally in place of the artifact.

2447 **3.5.1 Element `<ArtifactResolve>`**

2448 The `<ArtifactResolve>` message is used to request that a SAML protocol message be returned in an
2449 `<ArtifactResponse>` message by specifying an artifact that represents the SAML protocol message.
2450 The original transmission of the artifact is governed by the specific protocol binding that is being used; see
2451 [SAMLBind] for more information on the use of artifacts in bindings.

2452 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
2453 protected by the protocol binding used to deliver the message.

2454 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
2455 adds the following element:

2456 <Artifact> [Required]

2457 The artifact value that the requester received and now wishes to translate into the protocol message it
2458 represents. See [SAMLBind] for specific artifact format information.

2459 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**
2460 complex type:

```
2461 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
2462 <complexType name="ArtifactResolveType">
2463   <complexContent>
2464     <extension base="samlp:RequestAbstractType">
2465       <sequence>
2466         <element ref="samlp:Artifact"/>
2467       </sequence>
2468     </extension>
2469   </complexContent>
2470 </complexType>
2471 <element name="Artifact" type="string"/>
```

2472 3.5.2 Element <ArtifactResponse>

2473 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
2474 message element. This element is of complex type **ArtifactResponseType**, which extends
2475 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2476 message being returned. This wrapped message element can be a request or a response.

2477 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2478 protected by the protocol binding used to deliver the message.

2479 The following schema fragment defines the <ArtifactResponse> element and its
2480 **ArtifactResponseType** complex type:

```
2481 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2482 <complexType name="ArtifactResponseType">
2483   <complexContent>
2484     <extension base="samlp:StatusResponseType">
2485       <sequence>
2486         <any namespace="##any" processContents="lax" minOccurs="0"/>
2487       </sequence>
2488     </extension>
2489   </complexContent>
2490 </complexType>
```

2491 3.5.3 Processing Rules

2492 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2493 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>
2494 element with no embedded message. In both cases, the <Status> element MUST include a
2495 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A
2496 response message with no embedded message inside it is termed an empty response in the remainder of
2497 this section.

2498 The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent
2499 request with the same artifact by any requester results in an empty response as described above.

2500 Some SAML protocol messages, most particularly the `<AuthnRequest>` message in some profiles, MAY
2501 be intended for consumption by any party that receives it and can respond appropriately. In most other
2502 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2503 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2504 receives an `<ArtifactResolve>` message from a requester that cannot authenticate itself as the
2505 original intended recipient, then the artifact issuer MUST return an empty response.

2506 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2507 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2508 return it in an `<ArtifactResolve>` message to the issuer.

2509 Note that the `<ArtifactResponse>` message's `InResponseTo` attribute MUST contain the value of
2510 the corresponding `<ArtifactResolve>` message's `ID` attribute, but the embedded protocol message
2511 will contain its own message identifier, and in the case of an embedded response, may contain a different
2512 `InResponseTo` value that corresponds to the original request message to which the embedded message
2513 is responding.

2514 All other processing rules associated with the underlying request and response messages MUST be
2515 observed.

2516 **3.6 Name Identifier Management Protocol**

2517 After establishing a name identifier for a principal, an identity provider wishing to change the value
2518 [\[E12\]](#)and/or format of the identifier that it will use when referring to the principal, or to indicate that a name
2519 identifier will no longer be used to refer to the principal, informs service providers of the change by
2520 sending them a `<ManageNameIDRequest>` message.

2521 A service provider also uses this message to register or change the `SPProvidedID` value to be included
2522 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
2523 identifier between itself and the identity provider.

2524 [\[E14\]](#)Note that this protocol is typically not used with "transient" name identifiers, since their value is not
2525 intended to be managed on a long term basis. This protocol MUST NOT be used in conjunction with the
2526 [urn:oasis:names:tc:SAML:2.0:nameidformat:transient](#) `<NameID>` Format.

2527 **3.6.1 Element `<ManageNameIDRequest>`**

2528 A provider sends a `<ManageNameIDRequest>` message to inform the recipient of a changed name
2529 identifier or to indicate the termination of the use of a name identifier.

2530 The `<ManageNameIDRequest>` message SHOULD be signed or otherwise authenticated and integrity
2531 protected by the protocol binding used to deliver the message.

2532 This message has the complex type **ManageNameIDRequestType**, which extends
2533 **RequestAbstractType** and adds the following elements:

2534 `<saml:NameID>` or `<saml:EncryptedID>` [Required]

2535 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
2536 principal as currently recognized by the identity and service providers prior to this request. (For more
2537 information on these elements, see Section 2.2.)

2538 `<NewID>` or `<NewEncryptedID>` or `<Terminate>` [Required]

2539 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2540 requesting provider concerning this principal, or an indication that the use of the old identifier has
2541 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
2542 appear in subsequent `<NameID>` elements in the `SPProvidedID` attribute. If the requester is the
2543 identity provider, the new value will appear in subsequent `<NameID>` elements as the element's

2544 content. [\[E12\]](#)In either case, if the `<NewEncryptedID>` is used, its encrypted content is just a
2545 `<NewID>` element containing only the new value for the identifier (format and qualifiers cannot be
2546 changed once established).

2547 The following schema fragment defines the `<ManageNameIDRequest>` element and its
2548 **ManageNameIDRequestType** complex type:

```
2549 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2550 <complexType name="ManageNameIDRequestType">
2551   <complexContent>
2552     <extension base="samlp:RequestAbstractType">
2553       <sequence>
2554         <choice>
2555           <element ref="saml:NameID"/>
2556           <element ref="saml:EncryptedID"/>
2557         </choice>
2558         <choice>
2559           <element ref="samlp:NewID"/>
2560           <element ref="samlp:NewEncryptedID"/>
2561           <element ref="samlp:Terminate"/>
2562         </choice>
2563       </sequence>
2564     </extension>
2565   </complexContent>
2566 </complexType>
2567 <element name="NewID" type="string"/>
2568 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
2569 <element name="Terminate" type="samlp:TerminateType"/>
2570 <complexType name="TerminateType"/>
```

2571 3.6.2 Element `<ManageNameIDResponse>`

2572 The recipient of a `<ManageNameIDRequest>` message MUST respond with a
2573 `<ManageNameIDResponse>` message, which is of type **StatusResponseType** with no additional
2574 content.

2575 The `<ManageNameIDResponse>` message SHOULD be signed or otherwise authenticated and integrity
2576 protected by the protocol binding used to deliver the message.

2577 The following schema fragment defines the `<ManageNameIDResponse>` element:

```
2578 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

2579 3.6.3 Processing Rules

2580 If the request includes a `<saml:NameID>` (or encrypted version) that the recipient does not recognize,
2581 the responding provider MUST respond with an error `<Status>` and MAY respond with a second-level
2582 `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`.

2583 If the `<Terminate>` element is included in the request, the requesting provider is indicating that (in the
2584 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of
2585 an identity provider) it will no longer issue assertions to the service provider [\[E55\]](#)using that identifier about
2586 the principal. The receiving provider can perform any maintenance with the knowledge that the
2587 relationship represented by the name identifier has been terminated. [\[E8\]](#) In general it SHOULD NOT
2588 invalidate any active session(s) of the principal for whom the relationship has been terminated. If the
2589 receiving provider is an identity provider, it SHOULD NOT invalidate any active session(s) of the principal
2590 established with other service providers. A requesting provider MAY send a `<LogoutRequest>` message
2591 prior to initiating a name identifier termination by sending a `<ManageNameIDRequest>` message if that is
2592 the requesting provider's intent (e.g., the name identifier termination is initiated via an administrator who
2593 wished to terminate all user activity). The requesting provider MUST NOT send a `<LogoutRequest>`

2594 | ~~message after the <ManageNameIDRequest> message is sent. It can choose to invalidate the active-~~
2595 | ~~session(s) of a principal for whom a relationship has been terminated.~~

2596 | [E14] If the receiving provider is maintaining state associated with the name identifier, such as the value of
2597 | the identifier itself (in the case of a pair-wise identifier), an SPProvidedID value, the sender's consent to
2598 | the identifier's creation/use, etc., then the receiver can perform any maintenance with the knowledge that
2599 | the relationship represented by the name identifier has been terminated.

2600 | Any subsequent operations performed by the receiver on behalf of the sender regarding the principal (for
2601 | example, a subsequent <AuthnRequest>) SHOULD be carried out in a manner consistent with the
2602 | absence of any previous state.

2603 | Termination is potentially the cleanup step for any state management behavior triggered by the use of the
2604 | AllowCreate attribute in the Authentication Request protocol (see Section 3.4). Deployments that do not
2605 | make use of that attribute are likely to avoid the use of the <Terminate> element or would treat it as a
2606 | purely advisory matter.

2607 | Note that in most cases (a notable exception being the rules surrounding the SPProvidedID attribute),
2608 | there are no requirements on either identity providers or service providers regarding the creation or use of
2609 | persistent state. Therefore, no explicit behavior is mandated when the <Terminate> element is received.
2610 | However, if persistent state is present pertaining to the use of an identifier (such as if an SPProvidedID
2611 | attribute was attached), the <Terminate> element provides a clear indication that this state SHOULD be
2612 | deleted (or marked as obsolete in some fashion).

2613 | If the service provider requests that its identifier for the principal be changed by including a <NewID> (or
2614 | <NewEncryptedID>) element, the identity provider MUST include the element's content as the
2615 | SPProvidedID when subsequently communicating to the service provider ~~regarding this~~
2616 | ~~principal~~[E55]using the primary identifier.

2617 | If the identity provider requests that its identifier for the principal be changed by including a <NewID> (or
2618 | <NewEncryptedID>) element, the service provider MUST use the element's content as the
2619 | <saml:NameID> element content when subsequently communicating with the identity provider [E55]in
2620 | any case where the identifier being changed would have been used regarding this principal.

2621 | Note that neither, either, or both of the original and new identifier MAY be encrypted (using the
2622 | <EncryptedID> and <NewEncryptedID> elements).

2623 | In any case, the <saml:NameID> content in the request and its associated SPProvidedID attribute
2624 | MUST contain the most recent name identifier information established between the providers for the
2625 | principal.

2626 | In the case of an identifier with a Format of urn:oasis:names:tc:SAML:2.0:nameid-
2627 | format:persistent, the NameQualifier attribute MUST contain the unique identifier of the identity
2628 | provider that created the identifier. If the identifier was established between the identity provider and an
2629 | affiliation group of which the service provider is a member, then the SPNameQualifier attribute MUST
2630 | contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the
2631 | service provider. These attributes MAY be omitted if they would otherwise match the value of the
2632 | containing protocol message's <Issuer> element, but this is NOT RECOMMENDED due to the
2633 | opportunity for confusion.

2634 | Changes to these identifiers may take a potentially significant amount of time to propagate through the
2635 | systems at both the requester and the responder. Implementations might wish to allow each party to
2636 | accept either identifier for some period of time following the successful completion of a name identifier
2637 | change. Not doing so could result in the inability of the principal to access resources.

2638 | All other processing rules associated with the underlying request and response messages MUST be
2639 | observed.

2640 3.7 Single Logout Protocol

2641 The single logout protocol provides a message exchange protocol by which all sessions provided by a
2642 particular session authority are near-simultaneously terminated. The single logout protocol is used either
2643 when a principal logs out at a session participant or when the principal logs out directly at the
2644 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
2645 the logout event can be indicated through the `Reason` attribute.

2646
2647 The principal may have established authenticated sessions with both the session authority and individual
2648 session participants, based on assertions containing authentication statements supplied by the session
2649 authority.

2650
2651 When the principal invokes the single logout process at a session participant, the session participant
2652 MUST send a `<LogoutRequest>` message to the session authority that provided the assertion
2653 containing the authentication statement related to that session at the session participant.

2654
2655 When either the principal invokes a logout at the session authority, or a session participant sends a logout
2656 request to the session authority specifying that principal, the session authority SHOULD send a
2657 `<LogoutRequest>` message to each session participant to which it provided assertions containing
2658 authentication statements under its current session with the principal, with the exception of the session
2659 participant that sent the `<LogoutRequest>` message to the session authority. It SHOULD attempt to
2660 contact as many of these participants as it can using this protocol, terminate its own session with the
2661 principal, and finally return a `<LogoutResponse>` message to the requesting session participant, if any.

2662 3.7.1 Element `<LogoutRequest>`

2663 A session participant or session authority sends a `<LogoutRequest>` message to indicate that a session
2664 has been terminated.

2665 The `<LogoutRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2666 by the protocol binding used to deliver the message.

2667 This message has the complex type `LogoutRequestType`, which extends `RequestAbstractType` and
2668 adds the following elements and attributes:

2669 `NotOnOrAfter` [Optional]

2670 The time at which the request expires, after which the recipient may discard the message. The time
2671 value is encoded in UTC, as described in Section 1.3.3.

2672 `Reason` [Optional]

2673 An indication of the reason for the logout, in the form of a URI reference. [\[E10\] The Reason attribute](#)
2674 [is specified as a string in the schema. This specification further restricts the schema by requiring that](#)
2675 [the Reason attribute MUST be in the form of a URI reference.](#)

2676 `<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

2677 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as
2678 currently recognized by the identity and service providers prior to this request. (For more information
2679 on this element, see Section 2.2.)

2680 `<SessionIndex>` [Optional]

2681 [\[E38\] The index of the session between the principal identified by the <saml:BaseID>.](#)
2682 [<saml:NameID>, or <saml:EncryptedID> element, and the session authority. This must correlate](#)
2683 [to the SessionIndex attribute, if any, in the <saml:AuthnStatement> of the assertion used to](#)
2684 [establish the session that is being terminated. The identifier that indexes this session at the message-](#)
2685 [recipient.](#)

2686 The following schema fragment defines the <LogoutRequest> element and associated
2687 **LogoutRequestType** complex type:

```
2688 <element name="LogoutRequest" type="saml:LogoutRequestType"/>
2689 <complexType name="LogoutRequestType">
2690 <complexContent>
2691 <extension base="saml:RequestAbstractType">
2692 <sequence>
2693 <choice>
2694 <element ref="saml:BaseID"/>
2695 <element ref="saml:NameID"/>
2696 <element ref="saml:EncryptedID"/>
2697 </choice>
2698 <element ref="saml:SessionIndex" minOccurs="0"
2699 maxOccurs="unbounded"/>
2700 </sequence>
2701 <attribute name="Reason" type="string" use="optional"/>
2702 <attribute name="NotOnOrAfter" type="dateTime"
2703 use="optional"/>
2704 </extension>
2705 </complexContent>
2706 </complexType>
2707 <element name="SessionIndex" type="string"/>
```

2708 3.7.2 Element <LogoutResponse>

2709 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of
2710 type **StatusResponseType**, with no additional content specified.

2711 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
2712 protected by the protocol binding used to deliver the message.

2713 The following schema fragment defines the <LogoutResponse> element:

```
2714 <element name="LogoutResponse" type="saml:StatusResponseType"/>
```

2715 3.7.3 Processing Rules

2716 The message sender MAY use the Reason attribute to indicate the reason for sending the
2717 <LogoutRequest>. The following values are defined by this specification for use by all message
2718 senders; other values MAY be agreed on between participants:

2719 urn:oasis:names:tc:SAML:2.0:logout:user

2720 Specifies that the message is being sent because the principal wishes to terminate the indicated
2721 session.

2722 urn:oasis:names:tc:SAML:2.0:logout:admin

2723 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2724 session for that principal.

2725 All other processing rules associated with the underlying request and response messages MUST be
2726 observed.

2727 Additional processing rules are provided in the following sections.

2728 3.7.3.1 Session Participant Rules

2729 When a session participant receives a <LogoutRequest> message, the session participant MUST
2730 authenticate the message. If the sender is the authority that provided an assertion containing an
2731 authentication statement linked to the principal's current session, the session participant MUST invalidate

2732 the principal's session(s) referred to by the <saml:BaseID>, <saml:NameID>, or
2733 <saml:EncryptedID> element, and any <SessionIndex> elements supplied in the message. If no
2734 <SessionIndex> elements are supplied, then all sessions associated with the principal MUST be
2735 invalidated.

2736
2737 The session participant MUST apply the logout request message to any assertion that meets the following
2738 conditions, even if the assertion arrives after the logout request:

- 2739 • The subject of the assertion **strongly matches** the <saml:BaseID>, <saml:NameID>, or
2740 <saml:EncryptedID> element in the <LogoutRequest>, as defined in Section 3.3.4.
- 2741 • The SessionIndex attribute of one of the assertion's authentication statements matches one of
2742 the <SessionIndex> elements specified in the logout request, or the logout request contains no
2743 <SessionIndex> elements.
- 2744 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself
2745 (in particular, the value of any specified NotOnOrAfter attributes in conditions or subject
2746 confirmation data).
- 2747 • The logout request has not yet expired (determined by examining the NotOnOrAfter attribute on
2748 the message).

2749 **Note:** This rule is intended to prevent a situation in which a session participant receives a
2750 logout request targeted at a single, or multiple, assertion(s) (as identified by the
2751 <SessionIndex> element(s)) *before* it receives the actual – and possibly still valid –
2752 assertion(s) targeted by the logout request. It should honor the logout request until the
2753 logout request itself may be discarded (the NotOnOrAfter value on the request has
2754 been exceeded) or the assertion targeted by the logout request has been received and
2755 has been handled appropriately.

2756 3.7.3.2 Session Authority Rules

2757 When a session authority receives a <LogoutRequest> message, the session authority MUST
2758 authenticate the sender. If the sender is a session participant to which the session authority provided an
2759 assertion containing an authentication statement for the current session, then the session authority
2760 SHOULD do the following in the specified order:

- 2761 • Send a <LogoutRequest> message to any session authority on behalf of whom the session
2762 authority proxied the principal's authentication, unless the second authority is the originator of the
2763 <LogoutRequest>.
- 2764 • Send a <LogoutRequest> message to each session participant for which the session authority
2765 provided assertions in the current session, *other than* the originator of a current
2766 <LogoutRequest>.
- 2767 • Terminate the principal's current session as specified by the <saml:BaseID>, <saml:NameID>,
2768 or <saml:EncryptedID> element, and any <SessionIndex> elements present in the logout
2769 request message.

2770 If the session authority successfully terminates the principal's session with respect to itself, then it MUST
2771 respond to the original requester, if any, with a <LogoutResponse> message containing a top-level
2772 status code of urn:oasis:names:tc:SAML:2.0:status:Success. If it cannot do so, then it MUST
2773 respond with a <LogoutResponse> message containing a top-level status code indicating the error.
2774 Thus, the top-level status indicates the state of the logout operation only with respect to the session
2775 authority itself.

2776 The session authority SHOULD attempt to contact each session participant using any applicable/usable
2777 protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because

2778 the original request takes place using a protocol binding that does not enable the logout to be propagated
2779 to all participants).

2780 In the event that not all session participants successfully respond to these <LogoutRequest> messages
2781 (or if not all participants can be contacted), then the session authority MUST include in its
2782 <LogoutResponse> message a second-level status code of
2783 urn:oasis:names:tc:SAML:2.0:status:PartialLogout to indicate that not all other session
2784 participants successfully responded with confirmation of the logout.

2785 Note that a session authority MAY initiate a logout for reasons other than having received a
2786 <LogoutRequest> from a session participant – these include, but are not limited to:

- 2787 • If some timeout period was agreed out-of-band with an individual session participant, the session
2788 authority MAY send a <LogoutRequest> to that individual participant alone.
- 2789 • An agreed global timeout period has been exceeded.
- 2790 • The principal or some other trusted entity has requested logout of the principal directly at the session
2791 authority.
- 2792 • The session authority has determined that the principal's credentials may have been compromised.

2793 When constructing a logout request message, the session authority MUST set the value of the
2794 NotOnOrAfter attribute of the message to a time value, indicating an expiration time for the message,
2795 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time
2796 value equal to or greater than the value of any NotOnOrAfter attribute specified in the assertion most
2797 recently issued as part of the targeted session (as indicated by the SessionIndex attribute on the logout
2798 request).

2799 In addition to the values specified in Section [\[E0\] 3.7.33-6-3](#) for the Reason attribute, the following values
2800 are also available for use by the session authority only:

2801 urn:oasis:names:tc:SAML:2.0:logout:global-timeout

2802 Specifies that the message is being sent because of the global session timeout interval period
2803 being exceeded.

2804 urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2805 Specifies that the message is being sent because a timeout interval period agreed between a
2806 participant and the session authority has been exceeded.

2807 **3.8 Name Identifier Mapping Protocol**

2808 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2809 identifier for the same principal in a particular format or federation namespace, it can send a request to
2810 the identity provider using this protocol.

2811 For example, a service provider that wishes to communicate with another service provider with whom it
2812 does not share an identifier for the principal can use an identity provider that shares an identifier for the
2813 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,
2814 with which it can communicate with the second service provider.

2815 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
2816 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

2817 3.8.1 Element <NameIDMappingRequest>

2818 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2819 <NameIDMappingRequest> message. This message has the complex type
2820 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2821 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2822 The identifier and associated descriptive data that specify the principal as currently recognized by the
2823 requester and the responder. (For more information on this element, see Section 2.2.)

2824 <NameIDPolicy> [Required]

2825 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2826 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2827 binding used to deliver the message.

2828 The following schema fragment defines the <NameIDMappingRequest> element and its
2829 **NameIDMappingRequestType** complex type:

```
2830 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2831 <complexType name="NameIDMappingRequestType">
2832   <complexContent>
2833     <extension base="samlp:RequestAbstractType">
2834       <sequence>
2835         <choice>
2836           <element ref="saml:BaseID"/>
2837           <element ref="saml:NameID"/>
2838           <element ref="saml:EncryptedID"/>
2839         </choice>
2840         <element ref="samlp:NameIDPolicy"/>
2841       </sequence>
2842     </extension>
2843   </complexContent>
2844 </complexType>
```

2845 3.8.2 Element <NameIDMappingResponse>

2846 The recipient of a <NameIDMappingRequest> message MUST respond with a
2847 <NameIDMappingResponse> message. This message has the complex type
2848 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

2849 <saml:NameID> or <saml:EncryptedID> [Required]

2850 The identifier and associated attributes that specify the principal in the manner requested, usually in
2851 encrypted form. (For more information on this element, see Section 2.2.)

2852 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2853 binding used to deliver the message.

2854 The following schema fragment defines the <NameIDMappingResponse> element and its
2855 **NameIDMappingResponseType** complex type:

```
2856 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2857 <complexType name="NameIDMappingResponseType">
2858   <complexContent>
2859     <extension base="samlp:StatusResponseType">
2860       <choice>
2861         <element ref="saml:NameID"/>
2862         <element ref="saml:EncryptedID"/>
2863       </choice>
2864     </extension>
```

2865 </complexContent>
2866 </complexType>

2867 3.8.3 Processing Rules

2868 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2869 <Status>. [E65]optionally containing a second-level <StatusCode> of
2870 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2871 At the responder's discretion, the
2872 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to
2873 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2874 All other processing rules associated with the underlying request and response messages MUST be
2875 observed.

2876 4 SAML Versioning

2877 The SAML specification set is versioned in two independent ways. Each is discussed in the following
2878 sections, along with processing rules for detecting and handling version differences. Also included are
2879 guidelines on when and why specific version information is expected to change in future revisions of the
2880 specification.

2881 When version information is expressed as both a Major and Minor version, it is expressed in the form
2882 *Major.Minor*. The version number *Major_B.Minor_B* is higher than the version number *Major_A.Minor_A* if and
2883 only if:

2884 $(Major_B > Major_A) \text{ OR } ((Major_B = Major_A) \text{ AND } (Minor_B > Minor_A))$

2885 4.1 SAML Specification Set Version

2886 Each release of the SAML specification set will contain a major and minor version designation describing
2887 its relationship to earlier and later versions of the specification set. The version will be expressed in the
2888 content and filenames of published materials, including the specification set documents and XML schema
2889 documents. There are no normative processing rules surrounding specification set versioning, since it
2890 merely encompasses the collective release of normative specification documents which themselves
2891 contain processing rules.

2892 The overall size and scope of changes to the specification set documents will informally dictate whether a
2893 set of changes constitutes a major or minor revision. In general, if the specification set is backwards
2894 compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain
2895 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
2896 revision.

2897 4.1.1 Schema Version

2898 As a non-normative documentation mechanism, any XML schema documents published as part of the
2899 specification set will contain a `version` attribute on the `<xs:schema>` element whose value is in the
2900 form *Major.Minor*, reflecting the specification set version in which it has been published. Validating
2901 implementations MAY use the attribute as a means of distinguishing which version of a schema is being
2902 used to validate messages, or to support multiple versions of the same logical schema.

2903 4.1.2 SAML Assertion Version

2904 The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the
2905 assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed
2906 so as to document the syntax, semantics, and processing rules of the assertions of the same version.
2907 That is, specification set version 1.0 describes assertion version 1.0, and so on.

2908 There is explicitly NO relationship between the assertion version and the target XML namespace specified
2909 for the schema definitions for that assertion version.

2910 The following processing rules apply:

- 2911 • A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion
2912 version number not supported by the authority.
- 2913 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
2914 supported by the relying party.
- 2915 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version
2916 number is higher than the minor assertion version number supported by the relying party. However,
2917 all assertions that share a major assertion version number MUST share the same general

2918 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For
2919 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the
2920 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the
2921 likely effects of schema evolution.)

2922 **4.1.3 SAML Protocol Version**

2923 The various SAML protocols' request and response elements contain an attribute for expressing the major
2924 and minor version of the request or response message using a string of the form *Major.Minor*. Each
2925 version of the SAML specification set will be construed so as to document the syntax, semantics, and
2926 processing rules of the protocol messages of the same version. That is, specification set version 1.0
2927 describes request and response version V1.0, and so on.

2928 There is explicitly NO relationship between the protocol version and the target XML namespace specified
2929 for the schema definitions for that protocol version.

2930 The version numbers used in SAML protocol request and response elements will match for any particular
2931 revision of the SAML specification set.

2932 **4.1.3.1 Request Version**

2933 The following processing rules apply to requests:

- 2934 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2935 SAML requester and the SAML responder.
- 2936 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD
2937 assume that the responder supports requests with the highest request version supported by the
2938 requester.
- 2939 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version
2940 number matching a response version number that the requester does not support.
- 2941 • A SAML responder MUST reject any request with a major request version number not supported by
2942 the responder.
- 2943 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2944 higher than the highest supported request version that it supports. However, all requests that share
2945 a major request version number MUST share the same general processing rules and semantics,
2946 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the
2947 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill
2948 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

2949 **4.1.3.2 Response Version**

2950 The following processing rules apply to responses:

- 2951 • A SAML responder MUST NOT issue a response message with a response version number higher
2952 than the request version number of the corresponding request message.
- 2953 • A SAML responder MUST NOT issue a response message with a major response version number
2954 lower than the major request version number of the corresponding request message except to
2955 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2956 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
2957 top-level `<StatusCode>` value of
2958 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting
2959 one of the following second-level values:
2960 `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`,

2961 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or
2962 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

2963 **4.1.3.3 Permissible Version Combinations**

2964 Assertions of a particular major version appear only in response messages of the same major version, as
2965 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2966 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2967 response message, if the appropriate assertion schema is referenced during namespace importation. But
2968 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major
2969 versions.

2970 **4.2 SAML Namespace Version**

2971 XML schema documents published as part of the specification set contain one or more target
2972 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct
2973 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that
2974 part of the specification.

2975 The namespace URI references defined by the specification set will generally contain version information
2976 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond
2977 to the major and minor version of the specification set in which the namespace is first introduced and
2978 defined. This information is not typically consumed by an XML processor, which treats the namespace
2979 opaquely, but is intended to communicate the relationship between the specification set and the
2980 namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that are
2981 listed in Section 8.

2982 As a general rule, implementers can expect the namespaces and the associated schema definitions
2983 defined by a major revision of the specification set to remain valid and stable across minor revisions of the
2984 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but
2985 this is expected to be rare. In such cases, the older namespaces and their associated definitions should
2986 be expected to remain valid until a major specification set revision.

2987 **4.2.1 Schema Evolution**

2988 In general, maintaining namespace stability while adding or changing the content of a schema are
2989 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2990 older implementations will react to any given change, making forward compatibility difficult to achieve.
2991 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
2992 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),
2993 implementations should expect forward-compatible schema changes in minor revisions, allowing new
2994 messages to validate against older schemas.

2995 Implementations SHOULD expect and be prepared to deal with new extensions and message types in
2996 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types
2997 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such
2998 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
2999 include new query, statement, or condition types.

3000 5 SAML and XML Signature Syntax and Processing

3001 SAML assertions and SAML protocol request and response messages may be signed, with the following
3002 benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the
3003 asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-
3004 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the
3005 message originator supports message integrity, authentication of message origin to a destination, and, if
3006 the signature is based on the originator's public-private key pair, non-repudiation of origin.

3007 A digital signature is not always required in SAML. For example, in some circumstances, signatures may
3008 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing
3009 protocol response message. "Inherited" signatures should be used with care when the contained object
3010 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context
3011 must be retained to allow validation, exposing the XML content and adding potentially unnecessary
3012 overhead. As another example, the SAML relying party or SAML requester may have obtained an
3013 assertion or protocol message from the SAML asserting party or SAML responder directly (with no
3014 intermediaries) through a secure channel, with the asserting party or SAML responder having
3015 authenticated to the relying party or SAML responder by some means other than a digital signature.

3016 Many different techniques are available for "direct" authentication and secure channel establishment
3017 between two parties. The list includes TLS/SSL (see [RFC 2246]/[SSL3]), HMAC, password-based
3018 mechanisms, and so on. In addition, the applicable security requirements depend on the communicating
3019 applications and the nature of the assertion or message transported. It is RECOMMENDED that, in all
3020 other contexts, digital signatures be used for assertions and request and response messages.
3021 Specifically:

- 3022 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting
3023 party SHOULD be signed by the SAML asserting party.
- 3024 • A SAML protocol message arriving at a destination from an entity other than the originating sender
3025 SHOULD be signed by the sender.
- 3026 • Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that
3027 contain SAML documents. Caveats about retaining context and interoperability apply. XML
3028 Signatures are intended to be the primary SAML signature mechanism, but this specification
3029 attempts to ensure compatibility with profiles that may require other mechanisms.
- 3030 • Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be
3031 enveloped.

3032 5.1 Signing Assertions

3033 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as
3034 described in Section 2.

3035 5.2 Request/Response Signing

3036 All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected
3037 in the schema as described in Section 3.

3038 5.3 Signature Inheritance

3039 A SAML assertion may be embedded within another SAML element, such as an enclosing <Assertion>
3040 or a request or response, which may be signed. When a SAML assertion does not contain a
3041 <ds:Signature> element, but is contained in an enclosing SAML element that contains a
3042 <ds:Signature> element, and the signature applies to the <Assertion> element and all its children,

3043 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
3044 interpretation should be equivalent to the case where the assertion itself was signed with the same key
3045 and signature options.

3046 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
3047 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY
3048 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
3049 information from the surrounding context, but no such inheritance should be inferred unless specifically
3050 identified by the profile.

3051 **5.4 XML Signature Profile**

3052 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
3053 and many choices. This section details constraints on these facilities so that SAML processors do not
3054 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
3055 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**
3056 attribute on `<Assertion>` and the various request and response elements. These attributes are
3057 collectively referred to in this section as the identifier attributes.

3058 Note that this profile only applies to the use of the `<ds:Signature>` elements found directly within SAML
3059 assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to
3060 SAML content are free to define other approaches.

3061 **5.4.1 Signing Formats and Algorithms**

3062 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
3063 detached.

3064 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
3065 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
3066 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

3067 **5.4.2 References**

3068 SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of
3069 the assertion or protocol message being signed. The assertion's or protocol message's root element may
3070 or may not be the root element of the actual XML document containing the signed assertion or protocol
3071 message (e.g., it might be contained within a SOAP envelope).

3072 Signatures MUST contain a single `<ds:Reference>` containing a same-document reference to the **ID**
3073 attribute value of the root element of the assertion or protocol message being signed. For example, if the
3074 **ID** attribute value is "foo", then the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

3075 **5.4.3 Canonicalization Method**

3076 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
3077 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a
3078 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over
3079 SAML messages embedded in an XML context can be verified independent of that context.

3080 **5.4.4 Transforms**

3081 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
3082 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive

3083 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
3084 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

3085 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
3086 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can
3087 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
3088 applying the transforms manually to the content and reverifying the result as consisting of the same SAML
3089 message.

3090 5.4.5 KeyInfo

3091 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of
3092 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be
3093 absent.

3094 5.4.6 Example

3095 Following is an example of a signed response containing a signed assertion. Line breaks have been
3096 added for readability; the signatures are not valid and cannot be successfully verified.

```
3097 <Response
3098   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3099   ID="_c7055387-af61-4fce-8b98-e2927324b306"
3100   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
3101   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
3102   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
3103   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3104     <ds:SignedInfo>
3105       <ds:CanonicalizationMethod
3106         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3107       <ds:SignatureMethod
3108         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3109       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">
3110         <ds:Transforms>
3111           <ds:Transform
3112             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3113 signature" />
3114           <ds:Transform
3115             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3116             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
3117               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3118           </ds:Transform>
3119         </ds:Transforms>
3120         <ds:DigestMethod
3121           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
3122         <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
3123       </ds:Reference>
3124     </ds:SignedInfo>
3125     <ds:SignatureValue>
3126       x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwCRKrtwPS6vdVNCcY5rHaFPYwKf+5
3127       ELYcPzx+pXlh43SmwviCqXRjRtMANWbHLhWAptaKlywS7gFgsD0lqjyen3CP+m3D
3128       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
3129     </ds:SignatureValue>
3130     <ds:KeyInfo>
3131       <ds:X509Data>
3132         <ds:X509Certificate>
3133           MIICyJCCAajOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgaxCzAJBgNVBAYTA1VT
3134           MRlWEAyDVQQIEw1XaXNjb25zaW4xEDAoBgNVBACTB01hZGlzb24xIDAeBgNVBAoT
3135           F1VuaXZlcuNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLExJEaXZpc2lubiBvZiBJ
3136           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIiwYDVQDExIRVBLSSBTZXJ2ZXIgc0Eg
3137           Ls0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsxCzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
3138           CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
```



```
3139 Ym9yMQ4wDAYDVQKKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3140 dTEncUCUGCSqGSib3DQEJARYYcm9vdEBzaGlicMS5pbnRlcm5ldDIuZWZWR1MIGfMA0G
3141 CSqGSib3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
3142 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBiaOAPSZB113R6+KYiE7x4XAWIrcP+
3143 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
3144 pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3145 hkiG9w0BAQQFAA0BgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3146 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3147 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==
3148 </ds:X509Certificate>
3149 </ds:X509Data>
3150 </ds:KeyInfo>
3151 </ds:Signature>
3152 <Status>
3153 <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
3154 </Status>
3155 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3156 IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3157 xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3158 <Issuer>https://www.opensaml.org/IDP</Issuer>
3159 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3160 <ds:SignedInfo>
3161 <ds:CanonicalizationMethod
3162 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3163 <ds:SignatureMethod
3164 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3165 <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3166 <ds:Transforms>
3167 <ds:Transform
3168 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3169 signature"/>
3170 <ds:Transform
3171 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3172 <InclusiveNamespaces
3173 PrefixList="#default saml ds xs xsi"
3174 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3175 </ds:Transform>
3176 </ds:Transforms>
3177 <ds:DigestMethod
3178 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
3179 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3180 </ds:Reference>
3181 </ds:SignedInfo>
3182 <ds:SignatureValue>
3183 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgzPkJN9CMLG8ENR4Nrw+n
3184 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
3185 Mwul/cBUj20tBZQMFN7jQ9YB7klIz3RqVL+wNmeWI4=
3186 </ds:SignatureValue>
3187 <ds:KeyInfo>
3188 <ds:X509Data>
3189 <ds:X509Certificate>
3190 MIICyCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
3191 MRlWwEAYDVQQIEwI1XaXNjb25zaW4xEDA0BgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
3192 F1VuaXZlcnpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
3193 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXXJ2ZXIgc0Eg
3194 Ls0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVoowYsx
3195 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNNoaWdhbjESMBAGAlUEBxMJQW5uIEFy
3196 Ym9yMQ4wDAYDVQKKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3197 dTEncUCUGCSqGSib3DQEJARYYcm9vdEBzaGlicMS5pbnRlcm5ldDIuZWZWR1MIGfMA0G
3198 CSqGSib3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
3199 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBiaOAPSZB113R6+KYiE7x4XAWIrcP+
3200 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
3201 pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3202 hkiG9w0BAQQFAA0BgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3203 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3204 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==
```

```
3205         </ds:X509Certificate>
3206         </ds:X509Data>
3207     </ds:KeyInfo>
3208 </ds:Signature>
3209 <Subject>
3210     <NameID
3211         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3212         scott@example.org
3213     </NameID>
3214     <SubjectConfirmation
3215         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3216 </Subject>
3217 <Conditions NotBefore="2003-04-17T00:46:02Z"
3218     NotOnOrAfter="2003-04-17T00:51:02Z">
3219     <AudienceRestriction>
3220         <Audience>http://www.opensaml.org/SP</Audience>
3221     </AudienceRestriction>
3222 </Conditions>
3223 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3224     <AuthnContext>
3225         <AuthnContextClassRef>
3226             urn:oasis:names:tc:SAML:2.0:ac:classes:Password
3227         </AuthnContextClassRef>
3228     </AuthnContext>
3229 </AuthnStatement>
3230 </Assertion>
3231 </Response>
```

6 SAML and XML Encryption Syntax and Processing

3232

3233 Encryption is used as the means to implement confidentiality. The most common motives for
3234 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for
3235 competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness
3236 of some other security mechanism. For example, a secret password or key may be encrypted.

3237 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 3238 • Communications confidentiality may be provided by mechanisms associated with a particular
3239 binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS (see
3240 [RFC 2246]/[SSL3]) or SOAP Message Security mechanisms for confidentiality.
- 3241 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`
3242 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be
3243 encrypted.
- 3244 • An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- 3245 • The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- 3246 • An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

6.1 General Considerations

3247

3248 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use
3249 of XML Encryption [XMLEnc]. Encrypted data and [\[E30\]optionally zeroone](#) or more encrypted keys MUST
3250 replace the plaintext information in the same location within the XML instance. The `<EncryptedData>`
3251 element's `Type` attribute SHOULD be used and, if it is present, MUST have the value
3252 `http://www.w3.org/2001/04/xmlenc#Element`.

3253 Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The
3254 SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

6.2 ~~Combining Signatures and Encryption~~[\[E43\] Key and Data Referencing Guidelines](#)

3255

3256

~~3257 Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and~~
~~3258 encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in~~
~~3259 the reverse order that signing and encryption were performed.~~

- ~~3260 • When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and~~
~~3261 placed within the `<Assertion>` element before the element is encrypted.~~
- ~~3262 • When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be~~
~~3263 performed first and then the signature calculated over the assertion or message containing the~~
~~3264 encrypted element.~~

~~3265 [If an encrypted key is NOT included in the XML instance, then the relying party must be able to locally](#)~~
~~3266 [determine the decryption key, per \[XMLEnc\].](#)~~

~~3267 [Implementations of SAML MAY implicitly associate keys with the corresponding data they are used to](#)~~
~~3268 [encrypt, through the positioning of `<xenc:EncryptedKey>` elements next to the associated](#)~~
~~3269 [`<xenc:EncryptedData>` element, within the enclosing SAML parent element. However, the following](#)~~
~~3270 [set of explicit referencing guidelines are suggested to facilitate interoperability.](#)~~

~~3271 [If the encrypted key is included in the XML instance, then it SHOULD be referenced within the associated](#)~~
~~3272 [`<xenc:EncryptedData>` element, or alternatively embedded within the `<xenc:EncryptedData>`](#)~~

3273 [element. When an <xenc:EncryptedKey> element is used, the <ds:KeyInfo> element within](#)
 3274 [<xenc:EncryptedData> SHOULD reference the <xenc:EncryptedKey> element using a](#)
 3275 [<ds:RetrievalMethod> element of Type](#)
 3276 <http://www.w3.org/2001/04/xmlenc#EncryptedKey>.

3277 [In addition, an <xenc:EncryptedKey> element SHOULD contain an <xenc:ReferenceList>](#)
 3278 [element containing a <xenc:DataReference> that references the corresponding](#)
 3279 [<xenc:EncryptedData> element\(s\) that the key was used to encrypt.](#)

3280 [In scenarios where the encrypted element is being “multicast” to multiple recipients, and the key used to](#)
 3281 [encrypt the message must be in turn encrypted individually and independently for each of the multiple](#)
 3282 [recipients, the <xenc:CarriedKeyName> element SHOULD be used to assign a common name to each](#)
 3283 [of the <xenc:EncryptedKey> elements so that a <ds:KeyName> can be used from within the](#)
 3284 [<xenc:EncryptedData> element’s <ds:KeyInfo> element.](#)

3285 [Within the <xenc:EncryptedData> element, the <ds:KeyName> can be thought of as an “alias” that is](#)
 3286 [used for backwards referencing from the <xenc:CarriedKeyName> element in each individual](#)
 3287 [<xenc:EncryptedKey> element. While this accommodates a “multicast” approach, each recipient must](#)
 3288 [be able to understand \(at least one\) <ds:KeyName>. The Recipient attribute is used to provide a hint as](#)
 3289 [to which key is meant for which recipient.](#)

3290 [The SAML implementation has the discretion to accept or reject a message where multiple Recipient](#)
 3291 [attributes or <ds:KeyName> elements are understood. It is RECOMMENDED that implementations](#)
 3292 [simply use the first key they understand and ignore any additional keys.](#)

3293 **6.3 Examples**

3294 [In the following example, the parent element \(<EncryptedID>\) contains <xenc:EncryptedData> and](#)
 3295 [\(referenced\) <xenc:EncryptedKey> elements as siblings \(note that the key can in fact be anywhere in](#)
 3296 [the same instance, and the key references the <xenc:EncryptedData> element\):](#)

```

3297 <saml:EncryptedID xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
3298   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3299     Id="Encrypted_DATA_ID"
3300     Type="http://www.w3.org/2001/04/xmlenc#Element">
3301     <xenc:EncryptionMethod
3302       Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
3303     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3304       <ds:RetrievalMethod URI="#Encrypted_KEY_ID"
3305         Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
3306     </ds:KeyInfo>
3307     <xenc:CipherData>
3308       <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
3309     </xenc:CipherData>
3310   </xenc:EncryptedData>
3311
3312   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3313     Id="Encrypted_KEY_ID">
3314     <xenc:EncryptionMethod
3315       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3316     <xenc:CipherData>
3317       <xenc:CipherValue>PzA5X...</xenc:CipherValue>
3318     </xenc:CipherData>
3319     <xenc:ReferenceList>
3320       <xenc:DataReference URI="#Encrypted_DATA_ID"/>
3321     </xenc:ReferenceList>
3322   </xenc:EncryptedKey>
3323 </saml:EncryptedID>
  
```

3324 [In the following <EncryptedAttribute> example, the <xenc:EncryptedKey> element is contained](#)
 3325 [within the <xenc:EncryptedData> element, so there is no explicit referencing:](#)

```

3326 <saml:EncryptedAttribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
3327   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3328     Id="Encrypted_DATA_ID"
3329     Type="http://www.w3.org/2001/04/xmlenc#Element">
3330     <xenc:EncryptionMethod
3331       Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
3332     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3333       <xenc:EncryptedKey Id="Encrypted_KEY_ID">
3334         <xenc:EncryptionMethod
3335           Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3336         <xenc:CipherData>
3337           <xenc:CipherValue>SDFSDF... </xenc:CipherValue>
3338         </xenc:CipherData>
3339       </xenc:EncryptedKey>
3340     </ds:KeyInfo>
3341     <xenc:CipherData>
3342       <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
3343     </xenc:CipherData>
3344   </xenc:EncryptedData>
3345 </saml:EncryptedAttribute>

```

3346 [The final example shows an assertion encrypted for multiple recipients, using the](#)
3347 [<xenc:CarriedKeyName> approach:](#)

```

3348 <saml:EncryptedAssertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
3349   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3350     Id="Encrypted_DATA_ID"
3351     Type="http://www.w3.org/2001/04/xmlenc#Element">
3352     <xenc:EncryptionMethod
3353       Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
3354     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3355       <ds:KeyName>MULTICAST_KEY_NAME</ds:KeyName>
3356     </ds:KeyInfo>
3357     <xenc:CipherData>
3358       <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
3359     </xenc:CipherData>
3360   </xenc:EncryptedData>
3361
3362   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3363     Id="Encrypted_KEY_ID_1" Recipient="https://sp1.org">
3364     <xenc:EncryptionMethod
3365       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3366     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3367       <ds:KeyName>KEY_NAME_1</ds:KeyName>
3368     </ds:KeyInfo>
3369     <xenc:CipherData>
3370       <xenc:CipherValue>xyzABC...</xenc:CipherValue>
3371     </xenc:CipherData>
3372     <xenc:ReferenceList>
3373       <xenc:DataReference URI="#Encrypted_DATA_ID"/>
3374     </xenc:ReferenceList>
3375     <xenc:CarriedKeyName>MULTICAST_KEY_NAME</xenc:CarriedKeyName>
3376   </xenc:EncryptedKey>
3377
3378   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3379     Id="Encrypted_KEY_ID_2" Recipient="https://sp2.org">
3380     <xenc:EncryptionMethod
3381       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3382     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3383       <ds:KeyName>KEY_NAME_2</ds:KeyName>
3384     </ds:KeyInfo>
3385     <xenc:CipherData>
3386       <xenc:CipherValue>abcXYZ...</xenc:CipherValue>
3387     </xenc:CipherData>
3388     <xenc:ReferenceList>

```

```
3389 | <xenc:DataReference URI="#Encrypted_DATA_ID"/>
3390 | </xenc:ReferenceList>
3391 | <xenc:CarriedKeyName>MULTICAST_KEY_NAME</xenc:CarriedKeyName>
3392 | </xenc:EncryptedKey>
3393 | </saml:EncryptedAssertion>
```

3394 7 SAML Extensibility

3395 SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas.
3396 An example of an application that extends SAML assertions is the Liberty Protocols and Schema
3397 Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions
3398 and protocols.

3399 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can
3400 be combined with extensions to put the SAML framework to new uses.

3401 7.1 Schema Extension

3402 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
3403 elements can serve as the head element of a substitution group. However, SAML types are not defined as
3404 *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that
3405 extensions are typically defined only as types rather than elements, and are included in SAML instances
3406 by means of an `xsi:type` attribute.

3407 The following sections discuss only elements and types that have been specifically designed to support
3408 extensibility.

3409 7.1.1 Assertion Schema Extension

3410 The SAML assertion schema (see [SAML-XSD]) is designed to permit separate processing of the
3411 assertion package and the statements it contains, if the extension mechanism is used for either part.

3412 The following elements are intended specifically for use as extension points in an extension schema; their
3413 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 3414 • `<BaseID>` and **BaseIDAbstractType**
- 3415 • `<Condition>` and **ConditionAbstractType**
- 3416 • `<Statement>` and **StatementAbstractType**

- 3417 • The following constructs that are directly usable as part of SAML are particularly interesting targets for
3418 extension:
 - 3419 • `<AuthnStatement>` and **AuthnStatementType**
 - 3420 • `<AttributeStatement>` and **AttributeStatementType**
 - 3421 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
 - 3422 • `<AudienceRestriction>` and **AudienceRestrictionType**
 - 3423 • `<ProxyRestriction>` and **ProxyRestrictionType**
 - 3424 • `<OneTimeUse>` and **OneTimeUseType**

3425 7.1.2 Protocol Schema Extension

3426 The following SAML protocol [\[E61\]elementconstructs](#) are intended specifically for use as extension points
3427 in an extension schema; their types [listed](#) are set to *abstract*, and are thus usable only as the base of a
3428 derived type:

- 3429 • ~~`<Request>`~~ and **RequestAbstractType**
- 3430 • `<SubjectQuery>` and **SubjectQueryAbstractType**

3431 The following constructs that are directly usable as part of SAML are particularly interesting targets for
3432 extension:

- 3433 • <AuthnQuery> and **AuthnQueryType**
- 3434 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**
- 3435 • <AttributeQuery> and **AttributeQueryType**
- 3436 • **StatusResponseType**

3437 7.2 Schema Wildcard Extension Points

3438 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes
3439 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension
3440 schema.

3441 7.2.1 Assertion Extension Points

3442 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- 3443 • <SubjectConfirmationData>: Uses **xs:anyType**, which allows any sub-elements and
3444 attributes.
- 3445 • <AuthnContextDecl>: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3446 • <AttributeValue>: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3447 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other
3448 namespaces with lax schema validation processing.

3449 The following constructs in the assertion schema allow arbitrary global attributes:

- 3450 • <Attribute> and **AttributeType**

3451 7.2.2 Protocol Extension Points

3452 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- 3453 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema
3454 validation processing.
- 3455 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax
3456 schema validation processing.
- 3457 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with
3458 lax schema validation processing. (It is specifically intended to carry a SAML request or response
3459 message element, however.)

3460 7.3 Identifier Extension

3461 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
3462 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
3463 However, it is always possible to define additional URI-based identifiers for these purposes. It is
3464 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should
3465 the meaning of a given URI used as such an identifier significantly change, or be used to mean two
3466 different things.

3467 8 SAML-Defined Identifiers

3468 The following sections define URI-based identifiers for common resource access actions, subject name
3469 identifier formats, and attribute name formats.

3470 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of
3471 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
3472 have one of the following stems, according to the specification set version in which they were first
3473 introduced:

```
3474 urn:oasis:names:tc:SAML:1.0:  
3475 urn:oasis:names:tc:SAML:1.1:  
3476 urn:oasis:names:tc:SAML:2.0:
```

3477 8.1 Action Namespace Identifiers

3478 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to
3479 common sets of actions to perform on resources.

3480 8.1.1 Read/Write/Execute/Delete/Control

3481 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwdc`

3482 Defined actions:

3483 `Read Write Execute Delete Control`

3484 These actions are interpreted as follows:

3485 `Read`

3486 The subject may read the resource.

3487 `Write`

3488 The subject may modify the resource.

3489 `Execute`

3490 The subject may execute the resource.

3491 `Delete`

3492 The subject may delete the resource.

3493 `Control`

3494 The subject may specify the access control policy for the resource.

3495 8.1.2 Read/Write/Execute/Delete/Control with Negation

3496 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwdc-negation`

3497 Defined actions:

3498 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

3499 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions
3500 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
3501 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is
3502 affirmatively denied read permission.

3503 A SAML authority MUST NOT authorize both an action and its negated form.

3504 **8.1.3 Get/Head/Put/Post**

3505 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3506 Defined actions:

3507 GET HEAD PUT POST

3508 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
3509 the GET action on a resource is authorized to retrieve it.

3510 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST
3511 actions to the write permission. The correspondence is not exact however since an HTTP GET operation
3512 may cause data to be modified and a POST operation may cause modification to a resource other than
3513 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

3514 **8.1.4 UNIX File Permissions**

3515 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3516 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3517 The action string is a four-digit numeric code:

3518 *extended user group world*

3519 Where the *extended* access permission has the value

3520 +2 if sgid is set

3521 +4 if suid is set

3522 The *user group* and *world* access permissions have the value

3523 +1 if execute permission is granted

3524 +2 if write permission is granted

3525 +4 if read permission is granted

3526 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
3527 and execute; and world read.

3528 **8.2 Attribute Name Format Identifiers**

3529 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex
3530 type to refer to the classification of the attribute name for purposes of interpreting the name.

3531 **8.2.1 Unspecified**

3532 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3533 The interpretation of the attribute name is left to individual implementations.

3534 8.2.2 URI Reference

3535 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

3536 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML
3537 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-
3538 specific. See [SAMLProf] for attribute profiles that make use of this identifier.

3539 8.2.3 Basic

3540 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

3541 The class of strings acceptable as the attribute name MUST be drawn from the set of values belonging to
3542 the primitive type **xs:Name** as defined in [Schema2] Section 3.3.6. See [SAMLProf] for attribute profiles
3543 that make use of this identifier.

3544 8.3 Name Identifier Format Identifiers

3545 The following identifiers MAY be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
3546 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the
3547 associated processing rules, if any.

3548 **Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for
3549 SAML V2.0.

3550 8.3.1 Unspecified

3551 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3552 The interpretation of the content of the element is left to individual implementations.

3553 8.3.2 Email Address

3554 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3555 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
3556 defined in IETF RFC 2822 [RFC 2822] Section 3.4.1. An addr-spec has the form local-part@domain. Note
3557 that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded
3558 in parentheses) after it, and is not surrounded by "<" and ">".

3559 8.3.3 X.509 Subject Name

3560 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3561 Indicates that the content of the element is in the form specified for the contents of the
3562 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
3563 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
3564 differ from the rules given in IETF RFC 2253 [RFC 2253].

3565 8.3.4 Windows Domain Qualified Name

3566 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3567 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3568 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator
3569 MAY be omitted.

3570 **8.3.5 Kerberos Principal Name**

3571 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3572 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3573 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
3574 realm are described in IETF RFC 1510 [RFC 1510].

3575 **8.3.6 Entity Identifier**

3576 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3577 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3578 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service
3579 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to
3580 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make
3581 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be
3582 used in other elements and attributes whose purpose is to identify a system entity in various protocol
3583 exchanges.

3584 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3585 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3586 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

3587 **8.3.7 Persistent Identifier**

3588 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3589 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3590 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
3591 generated by identity providers MUST be constructed using pseudo-random values that have no
3592 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
3593 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
3594 Persistent name identifier values MUST NOT exceed a length of 256 characters. [\[E78\]A given value, once
3595 associated with a principal, MUST NOT be assigned to a different principal at any time in the future.](#)

3596 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity
3597 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived
3598 from the context of the message containing the element, such as the issuer of a protocol message or an
3599 assertion containing the identifier in its subject. Note that a different system entity might later issue its own
3600 protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in
3601 this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be
3602 omitted in such a case).

3603 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service
3604 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be
3605 omitted if the element is contained in a message intended only for consumption directly by the service
3606 provider, and the value would be the unique identifier of that service provider.

3607 ~~[\[E55\]The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most
3608 recently set by the service provider or affiliation, if any \(see Section 3.6\). If no such identifier has been
3609 established, then the attribute MUST be omitted.](#)~~

3610 Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be shared
3611 in clear text with providers other than the providers that have established the shared identifier.
3612 Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and
3613 protections. Deployments without such requirements are free to use other kinds of identifiers in their
3614 SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

3615 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3616 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3617 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3618 different and does not affect the behavior of the identity provider or any processing rules specific to
3619 persistent identifiers in the protocols defined in this specification.

3620 Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of
3621 creation, but not of use. If a persistent identifier is created by a particular identity provider, the
3622 `NameQualifier` attribute value is permanently established at that time. If a service provider that receives
3623 such an identifier takes on the role of an identity provider and issues its own assertion containing that
3624 identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It
3625 might alternatively choose to create its own persistent identifier to represent the principal and link the two
3626 values. This is a deployment decision.

3627 **8.3.8 Transient Identifier**

3628 **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

3629 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3630 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3631 accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of
3632 256 characters.

3633 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3634 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
3635 accordance with the rules specified in Section 8.3.7.

3636 **8.4 Consent Identifiers**

3637 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** and
3638 **StatusResponseType** complex types to communicate whether a principal gave consent, and under what
3639 conditions, for the message.

3640 **8.4.1 Unspecified**

3641 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

3642 No claim as to principal consent is being made.

3643 **8.4.2 Obtained**

3644 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

3645 Indicates that a principal's consent has been obtained by the issuer of the message.

3646 **8.4.3 Prior**

3647 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`

3648 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to
3649 the action that initiated the message.

3650 **8.4.4 Implicit**

3651 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3652 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the
3653 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically
3654 more proximal to the action in time and presentation than prior consent, such as part of a session of
3655 activities.

3656 **8.4.5 Explicit**

3657 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3658 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the
3659 action that initiated the message.

3660 **8.4.6 Unavailable**

3661 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3662 Indicates that the issuer of the message did not obtain consent.

3663 **8.4.7 Inapplicable**

3664 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3665 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

9 References

3666

3667 The following works are cited in the body of this specification.

9.1 Normative References

- 3669 **[Excl-C14N]** J. Boyer et al. *Exclusive XML Canonicalization Version 1.0*. World Wide Web Consortium, July 2002. See <http://www.w3.org/TR/xml-exc-c14n/>.
- 3670
- 3671 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- 3672
- 3673
- 3674 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-2/>.
- 3675
- 3676 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium, October 2000. See <http://www.w3.org/TR/REC-xml>.
- 3677
- 3678 **[XMLEnc]** D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>. Note that this specification normatively references [XMLEnc-XSD], listed below.
- 3679
- 3680
- 3681 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3682
- 3683 **[XMLNS]** T. Bray et al. *Namespaces in XML*. World Wide Web Consortium, January 1999. See <http://www.w3.org/TR/REC-xml-names>.
- 3684
- 3685 **[XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. [\[E74\]Second Edition](#). World Wide Web Consortium, [February 2002](#)[June 2008](#). See <http://www.w3.org/TR/xmlsig-core/>. Note that this specification normatively references [XMLSig-XSD], listed below.
- 3686
- 3687
- 3688
- 3689 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium. See <http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd>.
- 3690
- 3691

9.2 Non-Normative References

- 3692
- 3693 **[LibertyProt]** J. Beatty et al. *Liberty Protocols and Schema Specification Version 1.1*. Liberty Alliance Project, January 2003. See http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf.
- 3694
- 3695
- 3696
- 3697 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. See <http://www.ietf.org/rfc/rfc1510.txt>.
- 3698
- 3699 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 3700
- 3701 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See <http://www.ietf.org/rfc/rfc2246.txt>.
- 3702
- 3703 **[RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. See <http://www.ietf.org/rfc/rfc2253.txt>.
- 3704
- 3705
- 3706 **[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August, 1998. See <http://www.ietf.org/rfc/rfc2396.txt>.
- 3707
- 3708 **[RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. See <http://www.ietf.org/rfc/rfc2822.txt>.
- 3709

3710	[E74][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 .
3711		
3712	[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 .
3713		
3714	[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/ .
3715		
3716		
3717	[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/ .
3718		
3719		
3720	[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/ .
3721		
3722		
3723	[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/ .
3724		
3725		
3726	[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/ .
3727		
3728		
3729	[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/ .
3730		
3731		
3732	[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/ .
3733		
3734		
3735	[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/ .
3736		
3737		
3738		
3739	[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/ .
3740		
3741		
3742	[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/ .
3743		
3744		
3745	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
3746		
3747	[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 .
3748		
3749	[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 .
3750		
3751	[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/ .
3752		
3753		
3754	[XACML]	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See http://www.oasis-open.org/committees/xacml .
3755		
3756	[XML-ID]	J. Marsh et al. <i>xml:id Version 1.0</i> , World Wide Web Consortium, April 2004. See http://www.w3.org/TR/xml-id/ .
3757		

Appendix A. Acknowledgments

3759 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
3760 Committee, whose voting members at the time of publication were:

- 3761 • Conor Cahill, AOL
- 3762 • John Hughes, Atos Origin
- 3763 • Hal Lockhart, BEA Systems
- 3764 • Mike Beach, Boeing
- 3765 • Rebekah Metz, Booz Allen Hamilton
- 3766 • Rick Randall, Booz Allen Hamilton
- 3767 • Ronald Jacobson, Computer Associates
- 3768 • Gavenraj Sodhi, Computer Associates
- 3769 • Thomas Wisniewski, Entrust
- 3770 • Carolina Canales-Valenzuela, Ericsson
- 3771 • Dana Kaufman, Forum Systems
- 3772 • Irving Reid, Hewlett-Packard
- 3773 • Guy Denton, IBM
- 3774 • Heather Hinton, IBM
- 3775 • Maryann Hondo, IBM
- 3776 • Michael McIntosh, IBM
- 3777 • Anthony Nadalin, IBM
- 3778 • Nick Ragouzis, Individual
- 3779 • Scott Cantor, Internet2
- 3780 • Bob Morgan, Internet2
- 3781 • Peter Davis, Neustar
- 3782 • Jeff Hodges, Neustar
- 3783 • Frederick Hirsch, Nokia
- 3784 • Senthil Sengodan, Nokia
- 3785 • Abbie Barbir, Nortel Networks
- 3786 • Scott Kiestler, Novell
- 3787 • Cameron Morris, Novell
- 3788 • Paul Madsen, NTT
- 3789 • Steve Anderson, OpenNetwork
- 3790 • Ari Kermaier, Oracle
- 3791 • Vamsi Motukuru, Oracle
- 3792 • Darren Platt, Ping Identity
- 3793 • Prateek Mishra, Principal Identity
- 3794 • Jim Lien, RSA Security
- 3795 • John Linn, RSA Security
- 3796 • Rob Philpott, RSA Security
- 3797 • Dipak Chopra, SAP
- 3798 • Jahan Moreh, Sigaba
- 3799 • Bhavna Bhatnagar, Sun Microsystems

- 3800 • Eve Maler, Sun Microsystems
- 3801 • Ronald Monzillo, Sun Microsystems
- 3802 • Emily Xu, Sun Microsystems
- 3803 • Greg Whitehead, Trustgenix

3804

3805 The editors also would like to acknowledge the following former SSTC members for their contributions to
3806 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 3807 • Stephen Farrell, Baltimore Technologies
- 3808 • David Orchard, BEA Systems
- 3809 • Krishna Sankar, Cisco Systems
- 3810 • Zahid Ahmed, CommerceOne
- 3811 • Tim Alsop, CyberSafe Limited
- 3812 • Carlisle Adams, Entrust
- 3813 • Tim Moses, Entrust
- 3814 • Nigel Edwards, Hewlett-Packard
- 3815 • Joe Pato, Hewlett-Packard
- 3816 • Bob Blakley, IBM
- 3817 • Marlena Erdos, IBM
- 3818 • Marc Chanliau, Netegrity
- 3819 • Chris McLaren, Netegrity
- 3820 • Lynne Rosenthal, NIST
- 3821 • Mark Skall, NIST
- 3822 • Charles Knouse, Oblix
- 3823 • Simon Godik, Overxeer
- 3824 • Charles Norwood, SAIC
- 3825 • Evan Prodromou, Securant
- 3826 • Robert Griffin, RSA Security (former editor)
- 3827 • Sai Allarvarpu, Sun Microsystems
- 3828 • Gary Ellison, Sun Microsystems
- 3829 • Chris Ferris, Sun Microsystems
- 3830 • Mike Myers, Traceroute Security
- 3831 • Phillip Hallam-Baker, VeriSign (former editor)
- 3832 • James Vanderbeek, Vodafone
- 3833 • Mark O'Neill, Vordel
- 3834 • Tony Palmer, Vordel

3835

3836 Finally, the editors wish to acknowledge the following people for their contributions of material used as
3837 input to the OASIS Security Assertions Markup Language specifications:

- 3838 • Thomas Gross, IBM
- 3839 • Birgit Pfitzmann, IBM

3840 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on
3841 the SSTC was the primary editor of the errata working document and who made major substantive
3842 contributions to all of the errata materials.

3843 **Appendix B. Notices**

3844 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3845 might be claimed to pertain to the implementation or use of the technology described in this document or
3846 the extent to which any license under such rights might or might not be available; neither does it represent
3847 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
3848 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
3849 available for publication and any assurances of licenses to be made available, or the result of an attempt
3850 made to obtain a general license or permission for the use of such proprietary rights by implementors or
3851 users of this specification, can be obtained from the OASIS Executive Director.

3852 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
3853 other proprietary rights which may cover technology that may be required to implement this specification.
3854 Please address the information to the OASIS Executive Director.

3855 **Copyright © OASIS Open 2005. All Rights Reserved.**

3856 This document and translations of it may be copied and furnished to others, and derivative works that
3857 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
3858 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
3859 this paragraph are included on all such copies and derivative works. However, this document itself may
3860 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
3861 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
3862 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
3863 into languages other than English.

3864 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3865 or assigns.

3866 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3867 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3868 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3869 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.