



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

Last-Call Working Draft 17, 13 July 2004

Document identifier:

sstc-saml-core-2.0-draft-17

Location:

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

Editors:

Scott Cantor, Internet2 (cantor.2@osu.edu)
John Kemp, Nokia (john.kemp@nokia.com)
Eve Maler, Sun Microsystems (eve.maler@sun.com)

Contributors:

Stephen Farrell, Baltimore Technologies
Irving Reid, Baltimore Technologies
Hal Lockhart, BEA Systems
David Orchard, BEA Systems
Krishna Sankar, Cisco Systems
John Hughes, Entegriety
Carlisle Adams, Entrust
Tim Moses, Entrust
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Bob Blakley, IBM
Marlena Erdos, IBM
RL "Bob" Morgan, Internet2
Marc Chanliau, Netegrity
Chris McLaren, Netegrity
Prateek Mishra, Netegrity (co-chair)
Charles Knouse, Oblix
Simon Godik, Overxeer
John Linn, RSA Security
Rob Philpott, RSA Security (co-chair)
Darren Platt, formerly of RSA Security
Jahan Moreh, Sigaba
Jeff Hodges, Sun Microsystems
Phillip Hallam-Baker, VeriSign (former editor)

Abstract:

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

42 **Status:**

43 This is a last-call working draft produced by the Security Services Technical Committee. **See the**
44 **Revision History for details of changes made in this revision.**

45 Comments on this last-call draft are solicited by **2 August 2004** so that the TC can subsequently
46 prepare an OASIS Committee Draft. Committee members should submit comments and potential
47 errata to the security-services@lists.oasis-open.org list. Others should submit them by filling out
48 the web form located at [http://www.oasis-](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security)
49 [open.org/committees/comments/form.php?wg_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security). The committee will publish vetted
50 errata on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

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

55 Table of Contents

56	1 Introduction.....	7
57	1.1 Notation.....	7
58	1.2 Schema Organization and Namespaces.....	8
59	1.2.1 String and URI Values.....	8
60	1.2.2 Time Values.....	8
61	1.2.3 ID and ID Reference Values.....	8
62	1.2.4 Comparing SAML Values.....	9
63	2 SAML Assertions.....	10
64	2.1 Schema Header and Namespace Declarations.....	10
65	2.2 Name Identifiers.....	10
66	2.2.1 Element <BaseID>.....	11
67	2.2.2 Element <NameID>.....	11
68	2.2.3 Element <EncryptedID>.....	12
69	2.2.4 Element <Issuer>.....	12
70	2.3 Assertions.....	13
71	2.3.1 Element <AssertionIDRef>.....	13
72	2.3.2 Element <AssertionURIRef>.....	13
73	2.3.3 Element <Assertion>.....	13
74	2.3.3.1 Element <Subject>.....	15
75	2.3.3.2 Element <SubjectConfirmation>.....	15
76	2.3.3.3 Element <SubjectConfirmationData>.....	16
77	2.3.3.4 Complex Type KeyInfoConfirmationDataType.....	17
78	2.3.3.5 Element <Conditions>.....	17
79	2.3.3.5.1 Attributes NotBefore and NotOnOrAfter.....	18
80	2.3.3.5.2 Element <Condition>.....	19
81	2.3.3.5.3 Elements <AudienceRestriction> and <Audience>.....	19
82	2.3.3.5.4 Element <OneTimeUse>.....	19
83	2.3.3.5.5 Element <ProxyRestriction>.....	20
84	2.3.3.6 Element <Advice>.....	21
85	2.3.4 Element <EncryptedAssertion>.....	21
86	2.4 Statements.....	22
87	2.4.1 Element <Statement>.....	22
88	2.4.2 Element <AuthnStatement>.....	22
89	2.4.2.1 Element <SubjectLocality>.....	23
90	2.4.2.2 Element <AuthnContext>.....	24
91	2.4.3 Element <AttributeStatement>.....	25
92	2.4.3.1 Element <AttributeDesignator>.....	25
93	2.4.3.2 Element <Attribute>.....	26
94	2.4.3.2.1 Element <AttributeValue>.....	26
95	2.4.3.3 Element <EncryptedAttribute>.....	27
96	2.4.4 Element <AuthzDecisionStatement>.....	27
97	2.4.4.1 Simple Type DecisionType.....	29
98	2.4.4.2 Element <Action>.....	29
99	2.4.4.3 Element <Evidence>.....	30
100	3 SAML Protocols.....	31
101	3.1 Schema Header and Namespace Declarations.....	31
102	3.2 Requests and Responses.....	32
103	3.2.1 Complex Type RequestAbstractType.....	32
104	3.2.1.1 Complex Type StatusResponseType.....	33
105	3.2.1.2 Element <Status>.....	34

106	3.2.1.3 Element <StatusCode>.....	35
107	3.2.1.4 Element <StatusMessage>.....	37
108	3.2.1.5 Element <StatusDetail>.....	37
109	3.3 Assertion Query and Request Protocol.....	37
110	3.3.1 Element <AssertionIDRequest>.....	37
111	3.3.2 Queries.....	37
112	3.3.2.1 Element <SubjectQuery>.....	38
113	3.3.2.2 Element <AuthnQuery>.....	38
114	3.3.2.3 Element <RequestedAuthnContext>.....	39
115	3.3.2.4 Element <AttributeQuery>.....	40
116	3.3.2.5 Element <AuthzDecisionQuery>.....	40
117	3.3.3 Element <Response>.....	41
118	3.3.4 Processing Rules.....	42
119	3.4 Authentication Request Protocol.....	42
120	3.4.1 Element <AuthnRequest>.....	43
121	3.4.1.1 Element <NameIDPolicy>.....	45
122	3.4.1.2 Element <Scoping>.....	46
123	3.4.1.3 Element <IDPList>.....	46
124	3.4.1.4 Element <IDPEntry>.....	47
125	3.4.1.5 Processing Rules.....	47
126	3.4.1.6 Proxying.....	48
127	3.4.1.6.1 Proxying Processing Rules.....	48
128	3.5 Artifact Resolution Protocol.....	49
129	3.5.1 Element <ArtifactResolve>.....	50
130	3.5.2 Element <ArtifactResponse>.....	50
131	3.5.3 Processing Rules.....	51
132	3.6 Name Identifier Management Protocol.....	51
133	3.6.1 Element <ManageNameIDRequest>.....	52
134	3.6.2 Element <ManageNameIDResponse>.....	52
135	3.6.3 Processing Rules.....	53
136	3.7 Single Logout Protocol.....	53
137	3.7.1 Element <LogoutRequest>.....	54
138	3.7.2 Element <LogoutResponse>.....	55
139	3.7.3 Processing Rules.....	55
140	3.7.3.1 Session Participant Rules.....	55
141	3.7.3.2 Session Authority Rules.....	55
142	3.8 Name Identifier Mapping Protocol.....	56
143	3.8.1 Element <NameIDMappingRequest>.....	57
144	3.8.2 Element <NameIDMappingResponse>.....	57
145	3.8.3 Processing Rules.....	58
146	4 SAML Versioning.....	59
147	4.1 SAML Specification Set Version.....	59
148	4.1.1 Schema Version.....	59
149	4.1.2 SAML Assertion Version.....	59
150	4.1.3 SAML Protocol Version.....	60
151	4.1.3.1 Request Version.....	60
152	4.1.4 Response Version.....	60
153	4.1.5 Permissible Version Combinations.....	61
154	4.2 SAML Namespace Version.....	61
155	4.2.1 Schema Evolution.....	61
156	5 SAML and XML Signature Syntax and Processing.....	62
157	5.1 Signing Assertions.....	62
158	5.2 Request/Response Signing.....	62

159	5.3 Signature Inheritance.....	62
160	5.4 XML Signature Profile.....	63
161	5.4.1 Signing Formats and Algorithms.....	63
162	5.4.2 References.....	63
163	5.4.3 Canonicalization Method.....	63
164	5.4.4 Transforms.....	63
165	5.4.5 KeyInfo.....	64
166	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	64
167	5.4.7 Example.....	64
168	6 SAML and XML Encryption Syntax and Processing.....	67
169	6.1 General Considerations.....	67
170	6.2 Combining Signatures and Encyption.....	67
171	6.3 Examples.....	67
172	7 SAML Extensibility.....	68
173	7.1 Schema Extension.....	68
174	7.1.1 Assertion Schema Extension.....	68
175	7.1.2 Protocol Schema Extension.....	68
176	7.2 Schema Wildcard Extension Points.....	69
177	7.2.1 Assertion Extension Points.....	69
178	7.2.2 Protocol Extension Points.....	69
179	7.3 Identifier Extension.....	69
180	8 SAML-Defined Identifiers.....	70
181	8.1 Action Namespace Identifiers.....	70
182	8.1.1 Read/Write/Execute/Delete/Control.....	70
183	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	70
184	8.1.3 Get/Head/Put/Post.....	71
185	8.1.4 UNIX File Permissions.....	71
186	8.2 Attribute Name Format Identifiers.....	71
187	8.2.1 Unspecified.....	71
188	8.2.2 URI Reference.....	72
189	8.2.3 Basic.....	72
190	8.3 Name Identifier Format Identifiers.....	72
191	8.3.1 Unspecified.....	72
192	8.3.2 Email Address.....	72
193	8.3.3 X.509 Subject Name.....	72
194	8.3.4 Windows Domain Qualified Name.....	72
195	8.3.5 Kerberos Principal Name.....	73
196	8.3.6 Entity Identifier.....	73
197	8.3.7 Persistent Identifier.....	73
198	8.3.8 Transient Identifier.....	74
199	8.4 Consent Identifiers.....	74
200	8.4.1 Unspecified.....	74
201	8.4.2 Obtained.....	74
202	8.4.3 Prior.....	74
203	8.4.4 Implicit.....	74
204	8.4.5 Explicit.....	75
205	8.4.6 Unavailable.....	75
206	8.4.7 Inapplicable.....	75
207	9 References.....	76
208	9.1 Normative References.....	76
209	9.2 Non-Normative References.....	76

211 1 Introduction

212 This specification defines the syntax and semantics for Security Assertion Markup Language (SAML)
213 assertions and the protocols for requesting and returning them. SAML assertions, requests, and
214 responses are encoded in XML [XML] and use XML namespaces [XMLNS]. They are typically embedded
215 in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The
216 SAML specification for bindings [SAMLBind] provides frameworks for this embedding and transport. Files
217 containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAML-XSD] are
218 available. For general explanations of SAML terms and concepts, refer to the SAML technical overview
219 [SAML-TechOvw] and the SAML glossary [SAMLGloss].

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

221 1.1 Notation

222 This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative
223 text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages.

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

227 ...they MUST only be used where it is actually required for interoperation or to limit behavior
228 which has potential for causing harm (e.g., limiting retransmissions)...

229 These keywords are thus capitalized when used to unambiguously specify requirements over protocol and
230 application features and behavior that affect the interoperability and security of implementations. When
231 these words are not capitalized, they are meant in their natural-language sense.

232 Listings of SAML schemas appear like this.

233 Example code listings appear like this.

235 In cases of disagreement between the SAML schema documents [SAML-XSD] [SAML-XSD] and
236 schema listings in this specification, the schema documents take precedence. Note that in some cases
237 the normative text of this specification imposes constraints beyond those indicated by the schema
238 documents.

239 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for
240 their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is
241 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 [SAML-XSD]. The prefix is generally elided in mentions of XML protocol-related elements in text.
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].
xenc:	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax and Processing specification [XMLEnc] and its governing schema [XMLEnc-XSD].

Prefix	XML Namespace	Comments
xsd:	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. The prefix is generally shown in mentions of XML Schema-related constructs in text, however.

242 This specification uses the following typographical conventions in text: <SAML**E**lement>,
243 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword.

244 1.2 Schema Organization and Namespaces

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

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

248 The SAML request-response protocol structures are defined in a schema [SAML-P-XSD] associated with
249 the following XML namespace:

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

251 The assertion schema is imported into the protocol schema. Also imported into both schemas is the
252 schema for XML Signature [XMLSig], which is associated with the following XML namespace:

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

254 See Section 4.2 for information on SAML namespace versioning.

255 1.2.1 String and URI Values

256 All SAML string and URI reference values have the types **xsd:string** and **xsd:anyURI** respectively, which
257 are built in to the W3C XML Schema Datatypes specification [Schema2]. All strings in SAML messages
258 MUST consist of at least one non-whitespace character (whitespace is defined in the XML
259 Recommendation [XML] §2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise
260 indicated in this specification, all URI reference values MUST consist of at least one non-whitespace
261 character, and are REQUIRED to be absolute [RFC 2396].

262 1.2.2 Time Values

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

265 SAML system entities SHOULD NOT rely on other applications supporting time resolution finer than
266 milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

267 1.2.3 ID and ID Reference Values

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

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

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

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

287 1.2.4 Comparing SAML Values

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

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

299 Applications that compare data received in SAML documents to data from external sources MUST take
300 into account the normalization rules specified for XML. Text contained within elements is normalized so
301 that line endings are represented using linefeed characters (ASCII code 10_{Decimal}), as described in the XML
302 Recommendation [XML] §2.11. Attribute values defined as strings (or types derived from strings) are
303 normalized as described in [XML] §3.3.3. All whitespace characters are replaced with blanks (ASCII code
304 32_{Decimal}).

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

2 SAML Assertions

308

309 An assertion is a package of information that supplies one or more statements made by a SAML authority.
310 This SAML specification defines three different kinds of assertion statement that can be created by a
311 SAML authority. As described in Section 7, extensions are permitted by the SAML assertion schema,
312 allowing user-defined extensions to assertions and statements, as well as allowing the definition of new
313 kinds of assertion and statement. The three kinds of statement defined in this specification are:

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

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

2.1 Schema Header and Namespace Declarations

321

322 The following schema fragment defines the XML namespaces and other header information for the
323 assertion schema:

```
324 <schema
325   targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
326   xmlns="http://www.w3.org/2001/XMLSchema"
327   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
328   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
329   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
330   elementFormDefault="unqualified"
331   attributeFormDefault="unqualified"
332   blockDefault="substitution"
333   version="2.0">
334   <import namespace="http://www.w3.org/2000/09/xmldsig#"
335     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
336   schema.xsd"/>
337   <import namespace="http://www.w3.org/2001/04/xmlenc#"
338     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
339   schema.xsd"/>
340   <annotation>
341     <documentation>
342       Document identifier: sstc-saml-schema-assertion-2.0
343       Location: http://www.oasis-
344   open.org/committees/documents.php?wg\_abbrev=security
345     ...
346   </documentation>
347   </annotation>
348   ...
349 </schema>
```

2.2 Name Identifiers

350

351 The following sections define the SAML constructs that contain descriptive identifiers of subjects and
352 assertion and message issuers.

353 2.2.1 Element <BaseID>

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

357 NameQualifier [Optional]

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

360 SPNameQualifier [Optional]

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

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

```
365 <element name="BaseID" type="saml:BaseIDAbstractType"/>
366 <complexType name="BaseIDAbstractType" abstract="true" mixed="true">
367   <complexContent>
368     <extension base="anyType">
369       <attribute name="NameQualifier" type="string" use="optional"/>
370       <attribute name="SPNameQualifier" type="string" use="optional"/>
371     </extension>
372   </complexContent>
373 </complexType>
```

374 2.2.2 Element <NameID>

375 The <NameID> element is of type **NameIDType**, which restricts **BaseIDAbstractType** to simple string
376 content that contains the name identifier itself, that which provides additional attributes as follows:

377 Format [Optional]

378 A URI reference representing the classification of string-based identifier information. See Section
379 8.3 for some URI references that MAY be used as the value of the `Format` attribute and their
380 associated descriptions and processing rules. If no `Format` value is provided, the identifier
381 `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in
382 effect.

383 When a `Format` value other than those specified in Section 8.3 is used, the content of the
384 <NameID> element is to be interpreted according to the definition of that format as provided
385 outside of this specification. If not otherwise indicated by the definition of the format, issues of
386 anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and
387 relying parties are implementation-specific.

388 SPProvidedID [Optional]

389 A name identifier established by the service provider or affiliation of providers for the principal, if
390 different from the primary name identifier given in the content of the <NameID> element. This
391 attribute provides a means of integrating the use of SAML with existing identifiers already in use
392 by a service provider.

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

```
394 <element name="NameID" type="saml:NameIDType"/>
395 <complexType name="NameIDType" mixed="false">
```

```

396     <simpleContent>
397         <restriction base="saml:BaseIDAbstractType">
398             <simpleType>
399                 <restriction base="string"/>
400             </simpleType>
401             <attribute name="Format" type="anyURI" use="optional"/>
402             <attribute name="SPProvidedID" type="string" use="optional"/>
403         </restriction>
404     </simpleContent>
405 </complexType>

```

406 2.2.3 Element <EncryptedID>

407 The <EncryptedID> element extends **BaseIDAbstractType** to carry the content of the element in
408 encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. Its
409 <EncryptedID> element contains the following elements:

410 <xenc:EncryptedData> [Required]

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

416 <xenc:EncryptedKey> [Zero or More]

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

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

424 The following schema fragment defines the <EncryptedID> element and its **EncryptedIDType** complex
425 type:

```

426 <element name="EncryptedID" type="saml:EncryptedIDType"/>
427 <complexType name="EncryptedIDType" mixed="false">
428     <complexContent>
429         <restriction base="saml:BaseIDType">
430             <sequence>
431                 <element ref="xenc:EncryptedData"/>
432                 <element ref="xenc:EncryptedKey" minOccurs="0"
433 maxOccurs="unbounded"/>
434             </sequence>
435         </restriction>
436     </complexContent>
437 </complexType>

```

438 2.2.4 Element <Issuer>

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

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

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

445 2.3 Assertions

446 The following sections define the SAML constructs that contain assertion information.

447 2.3.1 Element <AssertionIDRef>

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

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

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

453 2.3.2 Element <AssertionURIRef>

454 The <AssertionURIRef> element makes a reference to a SAML assertion by URI reference. Resolving
455 the URI reference (in a fashion dictated by the URI reference itself) is intended to produce the assertion.
456 See the Bindings specification [SAMLBind] for information on how this element is used in a protocol
457 binding.

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

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

460 2.3.3 Element <Assertion>

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

463 **MajorVersion** [Required]

464 The major version of this assertion. The identifier for the version of SAML defined in this specification
465 is 2. SAML versioning is discussed in Section 4.

466 **MinorVersion** [Required]

467 The minor version of this assertion. The identifier for the version of SAML defined in this specification
468 is 0. SAML versioning is discussed in Section 4.

469 **ID** [Required]

470 The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in
471 Section 1.2.3 for identifier uniqueness.

472 **IssueInstant** [Required]

473 The time instant of issue in UTC, as described in Section 1.2.2.

474 **<Issuer>** [Required]

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

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

480 <ds:Signature> [Optional]

481 An XML Signature that authenticates the assertion, as described in Section 5.

482 <Subject> [Optional]

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

484 <Conditions> [Optional]

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

486 <Advice> [Optional]

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

489 Zero or more of the following statement elements:

490 <Statement>

491 A statement defined in an extension schema.

492 <AuthnStatement>

493 An authentication statement.

494 <AuthzDecisionStatement>

495 An authorization decision statement.

496 <AttributeStatement>

497 An attribute statement.

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

501 Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If omitted,
502 then the statements in the assertion are assumed to identify (implicitly or explicitly) the subject or subjects
503 to which they apply in an application- or profile-specific manner.

504 If a <ds:Signature> element is present, a relying party SHOULD verify that the signature is valid. If it is
505 invalid, the relying party SHOULD NOT rely on the contents of the assertion.

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

```
507 <element name="Assertion" type="saml:AssertionType"/>
508 <complexType name="AssertionType">
509   <sequence>
510     <element ref="saml:Issuer"/>
511     <element ref="ds:Signature" minOccurs="0"/>
512     <element ref="saml:Subject" minOccurs="0"/>
513     <element ref="saml:Conditions" minOccurs="0"/>
514     <element ref="saml:Advice" minOccurs="0"/>
515     <choice minOccurs="0" maxOccurs="unbounded">
516       <element ref="saml:Statement"/>
517       <element ref="saml:AuthnStatement"/>
518       <element ref="saml:AuthzDecisionStatement"/>
519       <element ref="saml:AttributeStatement"/>
520     </choice>
521   </sequence>
522   <attribute name="MajorVersion" type="integer" use="required"/>
523   <attribute name="MinorVersion" type="integer" use="required"/>
524   <attribute name="ID" type="ID" use="required"/>
525   <attribute name="IssueInstant" type="dateTime" use="required"/>
526 </complexType>
```

527 2.3.3.1 Element <Subject>

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

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

532 Identifies the subject.

533 <SubjectConfirmation> [Zero or More]

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

537 If the <Subject> element contains both an identifier and one or more subject confirmations, the SAML
538 authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it
539 can treat the entity presenting the assertion to the relying party as the entity that the SAML authority
540 associates with the name identifier for the purposes of processing the assertion. A <Subject> element
541 SHOULD NOT identify more than one principal. The following schema fragment defines the <Subject>
542 element and its **SubjectType** complex type:

```
543 <element name="Subject" type="saml:SubjectType"/>
544 <complexType name="SubjectType">
545   <choice>
546     <sequence>
547       <choice>
548         <element ref="saml:BaseID"/>
549         <element ref="saml:NameID"/>
550         <element ref="saml:EncryptedID"/>
551       </choice>
552       <element ref="saml:SubjectConfirmation" minOccurs="0"
553 maxOccurs="unbounded"/>
554     </sequence>
555     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
556   </choice>
557 </complexType>
```

558 2.3.3.2 Element <SubjectConfirmation>

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

562 Method [Required]

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

567 <SubjectConfirmationData> [Optional]

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

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

```

575 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
576 <complexType name="SubjectConfirmationType">
577   <sequence>
578
579     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
580   </sequence>
581   <attribute name="Method" type="anyURI" use="required"/>
582 </complexType>
583

```

584 2.3.3.3 Element <SubjectConfirmationData>

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

589 **NotBefore** [Optional]

590 A time instant before which the subject cannot be confirmed.

591 **NotOnOrAfter** [Optional]

592 A time instant at which the subject can no longer be confirmed.

593 **Recipient** [Optional]

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

595 **InResponseTo** [Optional]

596 Specifies the `RequestID` of a SAML protocol message in response to which an entity can present
597 the assertion while confirming itself.

598 **Address** [Optional]

599 Specifies the network address from which an entity can present the assertion while authenticating
600 itself.

601 Particular confirmation methods MAY require the use of one or more of these attributes. Note that the time
602 period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if any, SHOULD fall within the
603 overall assertion validity period as specified by the <Conditions> element's `NotBefore` and
604 `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore` MUST be less than
605 (earlier than) the value for `NotOnOrAfter`.

606 The following schema fragment defines the <SubjectConfirmationData> element and its
607 **SubjectConfirmationDataType** complex type:

```

608 <element name="SubjectConfirmationData"
609   type="saml:SubjectConfirmationDataType"/>
610 <complexType name="SubjectConfirmationDataType" mixed="true">
611   <complexContent>
612     <extension base="anyType">
613       <attribute name="NotBefore" type="dateTime"
614         use="optional"/>
615       <attribute name="NotOnOrAfter" type="dateTime"
616         use="optional"/>
617       <attribute name="Recipient" type="anyURI" use="optional"/>
618       <attribute name="InResponseTo" type="NCName"
619         use="optional"/>
620       <attribute name="Address" type="string" use="optional"/>
621     </extension>
622   </complexContent>
623 </complexType>

```

624 2.3.3.4 Complex Type **KeyInfoConfirmationDataType**

625 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`
626 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in
627 some way to authenticate the subject. The particular confirmation method MUST define the exact
628 mechanism by which the confirmation data can be used.

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

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

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

```
635 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
636   <complexContent>  
637     <restriction base="saml:SubjectConfirmationDataType">  
638       <sequence>  
639         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
640       </sequence>  
641     </restriction>  
642   </complexContent>  
643 </complexType>
```

644 2.3.3.5 Element **<Conditions>**

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

646 `NotBefore` [Optional]

647 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as
648 described in Section 1.2.2.

649 `NotOnOrAfter` [Optional]

650 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as
651 described in Section 1.2.2.

652 `<Condition>` [Any Number]

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

654 `<AudienceRestriction>` [Any Number]

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

656 `<OneTimeUse>` [Optional]

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

660 `<ProxyRestriction>` [Optional]

661 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent
662 assertions of their own on the basis of the information contained in the original assertion. Although the
663 schema permits multiple occurrences, there MUST be at most one instance of this element.

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

```
666 <element name="Conditions" type="saml:ConditionsType"/>
```

```

667 <complexType name="ConditionsType">
668   <choice minOccurs="0" maxOccurs="unbounded">
669     <element ref="saml:Condition"/>
670     <element ref="saml:AudienceRestriction"/>
671     <element ref="saml:OneTimeUse">
672     <element ref="saml:ProxyRestriction"/>
673   </choice>
674   <attribute name="NotBefore" type="dateTime" use="optional"/>
675   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
676 </complexType>

```

677 If an assertion contains a `<Conditions>` element, the validity of the assertion is dependent on the sub-
678 elements and attributes provided, using the following rules in the order shown:

- 679 1. If no sub-elements or attributes are supplied in the `<Conditions>` element, then the assertion is
680 considered to be **Valid**.
- 681 2. If any sub-element or attribute of the `<Conditions>` element is determined to be invalid, then the
682 assertion is considered to be **Invalid**.
- 683 3. If any sub-element or attribute of the `<Conditions>` element cannot be evaluated, then the validity
684 of the assertion cannot be determined and is considered to be **Indeterminate**.
- 685 4. If all sub-elements and attributes of the `<Conditions>` element are determined to be **Valid**, then the
686 assertion is considered to be **Valid**.

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

691 Note that an assertion that has validity status **Valid** may nonetheless be untrustworthy for reasons such as
692 not being issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

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

695 2.3.3.5.1 Attributes `NotBefore` and `NotOnOrAfter`

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

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

701 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
702 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the
703 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the
704 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),
705 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither
706 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is valid
707 at any time.

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

710 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is built
711 in to the W3C XML Schema Datatypes specification [Schema2]. All time instants are specified in Universal
712 Coordinated Time (UTC) as described in Section 1.2.2.

713 Implementations MUST NOT generate time instants that specify leap seconds.

714 2.3.3.5.2 Element <Condition>

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

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

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

721 2.3.3.5.3 Elements <AudienceRestriction> and <Audience>

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

727 <Audience>

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

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

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

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

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

```
743 <element name="AudienceRestriction"  
744 type="saml:AudienceRestrictionType"/>  
745 <complexType name="AudienceRestrictionType">  
746 <complexContent>  
747 <extension base="saml:ConditionAbstractType">  
748 <sequence>  
749 <element ref="saml:Audience" maxOccurs="unbounded"/>  
750 </sequence>  
751 </extension>  
752 </complexContent>  
753 </complexType>  
754 <element name="Audience" type="anyURI"/>
```

755 2.3.3.5.4 Element <OneTimeUse>

756 In general, relying parties may choose to retain assertions (or the information they contain in some other
757 form), and apply them repeatedly in making decisions. The <OneTimeUse> condition element allows an

758 authority to indicate that the information in the assertion is likely to change very soon and fresh information
759 should be obtained for each use. An example would be an assertion containing an
760 <AuthzDecisionStatement> which was the result of a policy which specified access control which
761 was a function of the time of day.

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

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

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

774 For the purposes of determining the validity of the <Conditions> element, the <OneTimeUse> is
775 considered to always be valid.

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

```
778 <element name="OneTimeUse" type="saml:OneTimeUseType"/>  
779 <complexType name="OneTimeUseType">  
780   <complexContent>  
781     <extension base="saml:ConditionAbstractType"/>  
782   </complexContent>  
783 </complexType>
```

784 **2.3.3.5.5 Element <ProxyRestriction>**

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

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

790 Count [Optional]

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

793 <Audience> [Zero or More]

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

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

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

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

805 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>
806 condition is considered to always be valid.

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

```
809 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
810 <complexType name="ProxyRestrictionType">
811   <complexContent>
812     <extension base="saml:ConditionAbstractType">
813       <sequence>
814         <element ref="saml:Audience" minOccurs="0"
815 maxOccurs="unbounded"/>
816       </sequence>
817       <attribute name="Count" type="nonNegativeInteger"
818 use="optional"/>
819     </extension>
820   </complexContent>
821 </complexType>
```

822 2.3.3.6 Element <Advice>

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

826 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,
827 <AssertionIDRef>, and <AssertionURIRef> elements, and elements in other namespaces, with
828 lax schema validation in effect for these other elements.

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

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

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

```
835 <element name="Advice" type="saml:AdviceType"/>
836 <complexType name="AdviceType">
837   <choice minOccurs="0" maxOccurs="unbounded">
838     <element ref="saml:AssertionIDRef"/>
839     <element ref="saml:AssertionURIRef"/>
840     <element ref="saml:Assertion"/>
841     <element ref="saml:EncryptedAssertion"/>
842     <any namespace="##other" processContents="lax"/>
843   </choice>
844 </complexType>
```

845 2.3.4 Element <EncryptedAssertion>

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

849 <xenc:EncryptedData> [Required]

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

854 <xenc:EncryptedKey> [Zero or More]

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

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

861 The following schema fragment defines the <EncryptedAssertion> element and its
862 **EncryptedAssertionType** complex type:

```
863 <element name="EncryptedAssertion" type="saml:EncryptedAssertionType"/>
864 <complexType name="EncryptedAssertionType">
865   <sequence>
866     <element ref="xenc:EncryptedData"/>
867     <element ref="xenc:EncryptedKey" minOccurs="0"
868 maxOccurs="unbounded"/>
869   </sequence>
870 </complexType>
```

871 2.4 Statements

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

873 2.4.1 Element <Statement>

874 The <Statement> element is an extension point that allows other assertion-based applications to reuse
875 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable
876 only as the base of a derived type.

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

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

881 2.4.2 Element <AuthnStatement>

882 The <AuthnStatement> element describes a statement by the SAML authority asserting that the
883 statement's subject was authenticated by a particular means at a particular time. It is of type
884 **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the following
885 elements and attributes:

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

888 `AuthnInstant` [Required]

889 Specifies the time at which the authentication took place. The time value is encoded in UTC, as

890 described in Section 1.2.2.

891 `SessionIndex` [Optional]

892 Indexes a particular session between the subject and the authority issuing this statement. The value
893 of the attribute SHOULD be a small, positive integer, but may be any string of text.

894 `SessionNotOnOrAfter` [Optional]

895 Specifies a time instant at which the session between the subject and the authority issuing this
896 statement MUST be considered ended. The time value is encoded in UTC, as described in Section
897 1.2.2.

898 `<SubjectLocality>` [Optional]

899 Specifies the DNS domain name and IP address for the system from which the subject was
900 apparently authenticated.

901 `<AuthnContext>` [Required]

902 The context used by the identity provider in the authentication event that yielded this statement.
903 Contains a reference to an authentication context class, an authentication context declaration,
904 declaration reference, or both. See the Authentication Context specification [SAMLAuthnCxt] for a full
905 description of authentication context information.

906 Assertions containing `<AuthnStatement>` elements MUST contain a `<Subject>`
907 element.

908 For privacy reasons, when including a `SessionIndex` attribute, the value MUST NOT be a unique value
909 identifying a principal's session at the authority. That is, it MUST NOT be reused in subsequent assertions
910 about the same principal. It MAY be a globally unique value such that it differs in each assertion (much as
911 the assertion's `ID` attribute does), or it MAY be a small integer value that is used in assertions issued on
912 behalf of many different subjects at the same time.

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

```
915 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
916 <complexType name="AuthnStatementType">
917   <complexContent>
918     <extension base="saml:StatementAbstractType">
919       <sequence>
920         <element ref="saml:SubjectLocality" minOccurs="0"/>
921         <element ref="saml:AuthnContext"/>
922       </sequence>
923       <attribute name="AuthnInstant" type="dateTime"
924 use="required"/>
925       <attribute name="SessionIndex" type="string"
926 use="optional"/>
927       <attribute name="SessionNotOnOrAfter" type="dateTime"
928 use="optional"/>
929     </extension>
930   </complexContent>
931 </complexType>
```

932 2.4.2.1 Element `<SubjectLocality>`

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

935 Address [Optional]

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

937 `DNSTName` [Optional]

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

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

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

```
943 <element name="SubjectLocality"
944         type="saml: SubjectLocalityType"/>
945 <complexType name="SubjectLocalityType">
946     <attribute name="Address" type="string" use="optional"/>
947     <attribute name="DNSTName" type="string" use="optional"/>
948 </complexType>
```

949 **2.4.2.2 Element `<AuthnContext>`**

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

953 `<AuthnContextClassRef>` [Optional]

954 A URI reference identifying an authentication context class that describes the authentication context
955 declaration that follows.

956 `<AuthnContextDecl>` or `<AuthnContextDeclRef>` [Optional]

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

960 `<AuthenticatingAuthority>` [Zero or More]

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

963 The following schema fragment defines the `<AuthnContext>` element and its **AuthnContextType**
964 complex type:

```
965 <element name="AuthnContext" type="saml:AuthnContextType"/>
966 <complexType name="AuthnContextType">
967     <sequence>
968         <element ref="saml:AuthnContextClassRef" minOccurs="0"/>
969         <choice minOccurs="0">
970             <element ref="saml:AuthnContextDecl"/>
971             <element ref="saml:AuthnContextDeclRef"/>
972         </choice>
973         <element ref="saml:AuthenticatingAuthority" minOccurs="0"
974 maxOccurs="unbounded"/>
975     </sequence>
976 </complexType>
977 <element name="AuthnContextClassRef" type="anyURI"/>
978 <element name="AuthnContextDeclRef" type="anyURI"/>
979 <element name="AuthnContextDecl" type="anyType"/>
980 <element name="AuthenticatingAuthority" type="anyURI"/>
```

981 2.4.3 Element <AttributeStatement>

982 The <AttributeStatement> element describes a statement by the SAML authority asserting that the
983 statement's subject is associated with the specified attributes. It is of type **AttributeStatementType**,
984 which extends **StatementAbstractType** with the addition of the following elements:

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

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

988 Assertions containing <AttributeStatement> elements MUST contain a <Subject> element.

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

```
991 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
992 <complexType name="AttributeStatementType">
993   <complexContent>
994     <extension base="saml:StatementAbstractType">
995       <choice maxOccurs="unbounded">
996         <element ref="saml:Attribute"/>
997         <element ref="saml:EncryptedAttribute"/>
998       </choice>
999     </extension>
1000   </complexContent>
1001 </complexType>
```

1002 2.4.3.1 Element <AttributeDesignator>

1003 The <AttributeDesignator> element identifies an attribute by name. It has the
1004 **AttributeDesignatorType** complex type. It is used in an attribute query to request that the values of
1005 specific SAML attributes be returned (see Section 3.3.2.4 for more information). The
1006 <AttributeDesignator> element contains the following XML attributes:

1007 Name [Required]

1008 The name of the attribute.

1009 NameFormat [Optional]

1010 A URI reference representing the classification of the attribute name for purposes of interpreting
1011 the name. See Section 8.3 for some URI references that MAY be used as the value of the
1012 NameFormat attribute and their associated descriptions and processing rules. If no NameFormat
1013 value is provided, the identifier urn:oasis:names:tc:SAML:2.0:attname-
1014 format:unspecified (see Section 8.3.1) is in effect.

1015 FriendlyName [Optional]

1016 A string that provides a more human-readable form of the attribute's name, which may be useful
1017 in cases in which the actual Name is complex, such as an OID or a UUID. This value MUST NOT
1018 be used as a basis for formally identifying SAML attributes.

1019 Arbitrary attributes

1020 This complex type uses an <xsd:anyAttribute> extension point to allow arbitrary XML
1021 attributes to be added to <AttributeDesignator> constructs without the need for an explicit
1022 schema extension. This allows additional fields to be added as needed to supply additional
1023 parameters to be used in an attribute query. SAML extensions MUST NOT add local (non-
1024 namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace
1025 to the **AttributeType** complex type or a derivation of it; such attributes are reserved for future
1026 maintenance and enhancement of SAML itself.

1027 The following schema fragment defines the <AttributeDesignator> element and its
1028 **AttributeDesignatorType** complex type:

```
1029 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
1030 <complexType name="AttributeDesignatorType">
1031   <attribute name="Name" type="string" use="required"/>
1032   <attribute name="NameFormat" type="anyURI" use="optional"/>
1033   <attribute name="FriendlyName" type="string" use="optional"/>
1034   <anyAttribute namespace="##other" processContents="lax"/>
1035 </complexType>
```

1036 2.4.3.2 Element <Attribute>

1037 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the
1038 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following
1039 element and attributes:

1040 <AttributeValue> [Any Number]

1041 The value of the attribute. If an attribute contains more than one discrete value, it is
1042 RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute
1043 exists but has no value, then the <AttributeValue> element MUST be omitted. If more than
1044 one <AttributeValue> element is supplied for an attribute, and any of the elements have a
1045 datatype assigned through `xsi:type`, then all of the <AttributeValue> elements must have
1046 the identical datatype assigned.

1047 Arbitrary attributes

1048 This complex type inherits from **AttributeDesignatorType** the ability to add arbitrary XML
1049 attributes to <Attribute> constructs without the need for an explicit schema extension. This
1050 allows additional fields to be added as needed to supply the context in which the attribute should
1051 be understood. SAML extensions MUST NOT add local (non-namespace-qualified) XML
1052 attributes or XML attributes qualified by a SAML-defined namespace to the **AttributeType**
1053 complex type or a derivation of it; such attributes are reserved for future maintenance and
1054 enhancement of SAML itself.

1055 The following schema fragment defines the <Attribute> element and its **AttributeType** complex type:

```
1056 <element name="Attribute" type="saml:AttributeType"/>
1057 <complexType name="AttributeType">
1058   <complexContent>
1059     <extension base="saml:AttributeDesignatorType">
1060       <sequence>
1061         <element ref="saml:AttributeValue" minOccurs="0"
1062 maxOccurs="unbounded"/>
1063       </sequence>
1064     </extension>
1065   </complexContent>
1066 </complexType>
```

1067 2.4.3.2.1 Element <AttributeValue>

1068 The <AttributeValue> element supplies the value of a specified attribute. It is of the **xsd:anyType**
1069 type, which allows any well-formed XML to appear as the content of the element.

1070 If the data content of an <AttributeValue> element is of an XML Schema simple type (such as
1071 **xsd:integer** or **xsd:string**), the datatype MAY be declared explicitly by means of an `xsi:type`
1072 declaration in the <AttributeValue> element. If the attribute value contains structured data, the
1073 necessary data elements MAY be defined in an extension schema.

1074 **Note:** Specifying a datatype on <AttributeValue> using `xsi:type` will require the
1075 presence of the extension schema that defines the datatype in order for schema
1076 processing to proceed.

1077 The following schema fragment defines the <AttributeValue> element:

```
1078 <element name="AttributeValue" type="anyType"/>
```

1079 2.4.3.3 Element <EncryptedAttribute>

1080 The <EncryptedAttribute> element represents a SAML attribute in encrypted fashion, as defined by
1081 the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAttribute>
1082 element contains the following elements:

1083 <xenc:EncryptedData> [Required]

1084 The encrypted content and associated encryption details, as defined by the XML Encryption
1085 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
1086 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
1087 encrypted content MUST contain an element that has a type that is derived from **AttributeType**.

1088 <xenc:EncryptedKey> [Zero or More]

1089 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
1090 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
1091 the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity as defined by
1092 Section 8.3.6.

1093 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through
1094 an intermediary.

1095 The following schema fragment defines the <EncryptedAttribute> element and its
1096 **EncryptedAttributeType** complex type:

```
1097 <element name="EncryptedAttribute" type="saml:EncryptedAttributeType"/>  
1098 <complexType name="EncryptedAttributeType">  
1099   <sequence>  
1100     <element ref="xenc:EncryptedData"/>  
1101     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>  
1102   </sequence>
```

1103 2.4.4 Element <AuthzDecisionStatement>

1104 **Note:** The <AuthzDecisionStatement> feature has been frozen as of SAML V2.0,
1105 with no future enhancements planned. Users who require additional functionality may
1106 want to consider the eXtensible Access Control Markup Language [XACML], which offers
1107 enhanced authorization decision features.

1108 The <AuthzDecisionStatement> element describes a statement by the SAML authority asserting that
1109 a request for access by the statement's subject to the specified resource has resulted in the specified
1110 authorization decision on the basis of some optionally specified evidence.

1111 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1112 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
1113 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
1114 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1115 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1116 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1117 dependent. When a scheme uses elements of the common syntax, it will also use the common
1118 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1119 with an explicit ".port", where the port is the default for the scheme, is equivalent to one where
1120 the port is elided.

1121 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the
1122 URI normalized form wherever possible as follows:

- 1123 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1124 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1125 Inconsistent URI reference interpretation can also result from differences between the URI reference
1126 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1127 employed to specify an access control policy language. The following security conditions SHOULD be
1128 satisfied by the system which employs SAML assertions:

- 1129 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
1130 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
1131 part of the resource URI reference.
- 1132 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
1133 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
1134 gain access to a denied resource by creating such an equivalence.

1135 The `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends
1136 **StatementAbstractType** with the addition of the following elements and attributes:

1137 `Resource` [Required]

1138 A URI reference identifying the resource to which access authorization is sought. This attribute MAY
1139 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the
1140 current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1141 `Decision` [Required]

1142 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1143 the **DecisionType** simple type.

1144 `<Action>` [One or more]

1145 The set of actions authorized to be performed on the specified resource.

1146 `<Evidence>` [Optional]

1147 A set of assertions that the SAML authority relied on in making the decision.

1148 Assertions containing `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

1149 The following schema fragment defines the `<AuthzDecisionStatement>` element and its
1150 **AuthzDecisionStatementType** complex type:

```
1151 <element name="AuthzDecisionStatement"  
1152 type="saml:AuthzDecisionStatementType"/>  
1153 <complexType name="AuthzDecisionStatementType">  
1154   <complexContent>  
1155     <extension base="saml:StatementAbstractType">  
1156       <sequence>  
1157         <element ref="saml:Action" maxOccurs="unbounded"/>  
1158         <element ref="saml:Evidence" minOccurs="0"/>  
1159       </sequence>  
1160       <attribute name="Resource" type="anyURI" use="required"/>  
1161     </extension>  
1162   </complexContent>  
1163 </complexType>
```

```

1161         <attribute name="Decision" type="saml:DecisionType"
1162 use="required"/>
1163     </extension>
1164 </complexContent>
1165 </complexType>

```

1166 2.4.4.1 Simple Type DecisionType

1167 The **DecisionType** simple type defines the possible values to be reported as the status of an
 1168 authorization decision statement.

1169 Permit

1170 The specified action is permitted.

1171 Deny

1172 The specified action is denied.

1173 Indeterminate

1174 The SAML authority cannot determine whether the specified action is permitted or denied.

1175 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to
 1176 provide an affirmative statement that it is not able to issue a decision. Additional information as to the
 1177 reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>` elements in
 1178 the enclosing `<Response>`.

1179 The following schema fragment defines the **DecisionType** simple type:

```

1180 <simpleType name="DecisionType">
1181     <restriction base="string">
1182         <enumeration value="Permit"/>
1183         <enumeration value="Deny"/>
1184         <enumeration value="Indeterminate"/>
1185     </restriction>
1186 </simpleType>

```

1187 2.4.4.2 Element <Action>

1188 The `<Action>` element specifies an action on the specified resource for which permission is sought. Its
 1189 string-data content provides the label for an action sought to be performed on the specified resource, and
 1190 it has the following attribute:

1191 Namespace [Optional]

1192 A URI reference representing the namespace in which the name of the specified action is to be
 1193 interpreted. If this element is absent, the namespace
 1194 `urn:oasis:names:tc:SAML:1.0:action:rwdc-negation` specified in Section 8.1.2 is in
 1195 effect.

1196 The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```

1197 <element name="Action" type="saml:ActionType"/>
1198 <complexType name="ActionType">
1199     <simpleContent>
1200         <extension base="string">
1201             <attribute name="Namespace" type="anyURI" use="required"/>
1202         </extension>
1203     </simpleContent>
1204 </complexType>

```

1205 **2.4.4.3 Element <Evidence>**

1206 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on
1207 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one
1208 or more of the following elements:

1209 <AssertionIDRef> [Any number]

1210 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1211 <AssertionURIRef> [Any number]

1212 Specifies an assertion by means of a URI reference.

1213 <Assertion> [Any number]

1214 Specifies an assertion by value.

1215 <EncryptedAssertion> [Any number]

1216 Specifies an encrypted assertion by value.

1217 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1218 and the SAML authority making the authorization decision. For example, in the case that the SAML relying
1219 party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1220 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1221 assertion as valid either to the relying party or any other third party.

1222 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
1223 <element name="Evidence" type="saml:EvidenceType"/>
1224 <complexType name="EvidenceType">
1225   <choice maxOccurs="unbounded">
1226     <element ref="saml:AssertionIDRef"/>
1227     <element ref="saml:AssertionURIRef"/>
1228     <element ref="saml:Assertion"/>
1229     <element ref="saml:EncryptedAssertion"/>
1230   </choice>
1231 </complexType>
```

3 SAML Protocols

1232

1233 SAML assertions and messages about them MAY be generated and exchanged using a variety of
1234 protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting
1235 queries, assertions, and messages using existing widely deployed transport protocols.

1236 Specific SAML request and response messages derive from common types. The requester sends an
1237 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1238 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1239



1241

Figure 1: SAML Request-Response Protocol

1242 The protocols defined by SAML achieve the following actions:

- 1243 • Returning one or more requested assertions (includes a direct request of the desired assertions, as
1244 well as querying for assertions that meet particular criteria)
- 1245 • Performing authentication on request and returning the corresponding assertion
- 1246 • Registering a name identifier or terminating a name registration on request
- 1247 • Retrieving a protocol message that has been requested by means of an artifact
- 1248 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
1249 request
- 1250 • Providing a name identifier mapping on request

1251 Throughout this section, text mentions of elements and types in the SAML protocol namespace are not
1252 shown with the conventional namespace prefix `samlp:`. For clarity, text mentions of elements and types
1253 in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:`.

3.1 Schema Header and Namespace Declarations

1254

1255 The following schema fragment defines the XML namespaces and other header information for the
1256 protocol schema:

```
1257 <schema  
1258   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1259   xmlns="http://www.w3.org/2001/XMLSchema"  
1260   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1261   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1262   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1263   elementFormDefault="unqualified"  
1264   attributeFormDefault="unqualified"  
1265   blockDefault="substitution"  
1266   version="2.0">  
1267   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
1268     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>  
1269   <import namespace="http://www.w3.org/2000/09/xmldsig#"
```

```

1270         schemaLocation=" http://www.w3.org/TR/xmlsig-core/xmlsig-core-
1271 schema.xsd"/>
1272         <annotation>
1273             <documentation>
1274                 Document identifier: sstc-saml-schema-protocol-2.0
1275                 Location: http://www.oasis-
1276 open.org/committees/documents.php?wg\_abbrev=security
1277                 ...
1278             </documentation>
1279         </annotation>
1280     ...
1281 </schema>

```

1282 3.2 Requests and Responses

1283 The following sections define the SAML constructs that underlie all of the request and response messages
1284 used in SAML protocols.

1285 3.2.1 Complex Type RequestAbstractType

1286 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1287 This type defines common attributes and elements that are associated with all SAML requests:

1288 **Note:** The `<RespondWith>` element has been removed from `<Request>` for V2.0 of
1289 SAML.

1290 `ID` [Required]

1291 An identifier for the request. It is of type `xsd:ID` and MUST follow the requirements specified in
1292 Section 1.2.3 for identifier uniqueness. The values of the `ID` attribute in a request and the
1293 `InResponseTo` attribute in the corresponding response MUST match.

1294 `MajorVersion` [Required]

1295 The major version of this request. The identifier for the version of SAML defined in this specification is
1296 2. SAML versioning is discussed in Section 4.

1297 `MinorVersion` [Required]

1298 The minor version of this request. The identifier for the version of SAML defined in this specification is
1299 0. SAML versioning is discussed in Section 4.

1300 `IssueInstant` [Required]

1301 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
1302 1.2.2.

1303 `Consent` [Optional]

1304 Indicates whether or not (and under what conditions) consent has been obtained from a user in the
1305 sending this request. See Section 8.4 for some URI references that MAY be used as the value of the
1306 `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the identifier
1307 `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in effect.

1308 `<saml:Issuer>` [Optional]

1309 Identifies the entity that generated the request message.

1310 `<ds:Signature>` [Optional]

1311 An XML Signature that authenticates the request, as described in Section 5.

1312 <Extensions> [Optional]

1313 This extension point contains optional protocol message extension elements that are agreed on
1314 between the communicating parties. No extension schema is required in order to make use of this
1315 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1316 for the extension to be valid. SAML extensions MUST NOT include local (non-namespace-qualified)
1317 elements or elements qualified by a SAML-defined namespace within this element.

1318 If a <ds:Signature> element is present, a responder SHOULD verify that the signature
1319 is valid. If it is invalid, the responder SHOULD NOT rely on the contents of the request
1320 and SHOULD respond with an error.

1321 If a Consent attribute is included and the value indicates that some form of user consent has been
1322 obtained, then the request SHOULD be signed.

1323 The following schema fragment defines the **RequestAbstractType** complex type:

```
1324 <complexType name="RequestAbstractType" abstract="true">  
1325   <sequence>  
1326     <element ref="saml:Issuer" minOccurs="0"/>  
1327     <element ref="ds:Signature" minOccurs="0"/>  
1328     <element ref="samlp:Extensions" minOccurs="0"/>  
1329   </sequence>  
1330   <attribute name="ID" type="ID" use="required"/>  
1331   <attribute name="MajorVersion" type="integer" use="required"/>  
1332   <attribute name="MinorVersion" type="integer" use="required"/>  
1333   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1334   <attribute name="Consent" type="anyURI" use="optional"/>  
1335 </complexType>  
1336 <element name="Extensions" type="samlp:ExtensionsType"/>  
1337 <complexType name="ExtensionsType">  
1338   <sequence>  
1339     <any namespace="##other" processContents="lax"  
1340     maxOccurs="unbounded"/>  
1341   </sequence>  
1342 </complexType>
```

1344 3.2.1.1 Complex Type StatusResponseType

1345 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
1346 defines common attributes and elements that are associated with all SAML responses:

1347 ID [Required]

1348 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in
1349 Section 1.2.3 for identifier uniqueness.

1350 InResponseTo [Optional]

1351 A reference to the identifier of the request to which the response corresponds, if any. If the response
1352 is not generated in response to a request, or if the ID attribute value of a request cannot be
1353 determined (because the request is malformed), then this attribute MUST NOT be present. Otherwise,
1354 it MUST be present and its value MUST match the value of the corresponding request's ID attribute
1355 value.

1356 MajorVersion [Required]

1357 The major version of this response. The identifier for the version of SAML defined in this specification
1358 is 2. SAML versioning is discussed in Section 4.

1359 MinorVersion [Required]

1360 The minor version of this response. The identifier for the version of SAML defined in this specification

1361 is 0. SAML versioning is discussed in Section 4.

1362 `IssueInstant` [Required]

1363 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
1364 1.2.2.

1365 `Recipient` [Optional]

1366 A URI reference indicating the intended recipient of this response. This is useful to prevent malicious
1367 forwarding of responses to unintended recipients, a protection that is required by some profiles of use.
1368 If it is present, the actual recipient MUST check that the URI reference identifies the recipient or a
1369 resource managed by the recipient. If it does not, the response MUST be discarded.

1370 `<saml:Issuer>` [Optional]

1371 Identifies the entity that generated the response message.

1372 `<ds:Signature>` [Optional]

1373 An XML Signature that authenticates the response, as described in Section 5.

1374 `<Extensions>` [Optional]

1375 This contains optional protocol message extension elements that are agreed on between the
1376 communicating parties. . No extension schema is required in order to make use of this extension
1377 point, and even if one is provided, the lax validation setting does not impose a requirement for the
1378 extension to be valid. SAML extensions MUST NOT include local (non-namespace-qualified)
1379 elements or elements qualified by a SAML-defined namespace within this element.

1380 `<Status>` [Required]

1381 A code representing the status of the corresponding request.

1382 If a `<ds:Signature>` element is present, a requester SHOULD verify that the signature is valid. If it is
1383 invalid, the requester SHOULD NOT rely on the contents of the response.

1384 The following schema fragment defines the **StatusResponseType** complex type:

```
1385 <complexType name="StatusResponseType">
1386   <sequence>
1387     <element ref="saml:Issuer" minOccurs="0"/>
1388     <element ref="ds:Signature" minOccurs="0"/>
1389     <element ref="samlp:Extensions" minOccurs="0"/>
1390     <element ref="samlp:Status"/>
1391   </sequence>
1392   <attribute name="ID" type="ID" use="required"/>
1393   <attribute name="InResponseTo" type="NCName" use="optional"/>
1394   <attribute name="MajorVersion" type="integer" use="required"/>
1395   <attribute name="MinorVersion" type="integer" use="required"/>
1396   <attribute name="IssueInstant" type="dateTime" use="required"/>
1397   <attribute name="Recipient" type="anyURI" use="optional"/>
1398 </complexType>
```

1399 3.2.1.2 Element `<Status>`

1400 The `<Status>` element contains the following elements:

1401 `<StatusCode>` [Required]

1402 A code representing the status of the corresponding request.

1403 `<StatusMessage>` [Optional]

1404 A message which MAY be returned to an operator.

1405 <StatusDetail> [Optional]
1406 Additional information concerning an error condition.

1407 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1408 <element name="Status" type="samlp:StatusType"/>  
1409 <complexType name="StatusType">  
1410   <sequence>  
1411     <element ref="samlp:StatusCode"/>  
1412     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1413     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1414   </sequence>  
1415 </complexType>
```

1416 3.2.1.3 Element <StatusCode>

1417 The <StatusCode> element specifies a code or a set of nested codes representing the status of the
1418 corresponding request. The <StatusCode> element has the following element and attribute:

1419 Value [Required]

1420 The status code value. This attribute contains a URI reference. The value of the topmost
1421 <StatusCode> element MUST be from the top-level list provided in this section.

1422 <StatusCode> [Optional]

1423 A subordinate status code that provides more specific information on an error condition.

1424 The permissible top-level <StatusCode> values are as follows:

1425 urn:oasis:names:tc:SAML:2.0:status:Success

1426 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1427 <StatusDetail> elements.

1428 urn:oasis:names:tc:SAML:2.0:status:Requester

1429 The request could not be performed due to an error on the part of the requester.

1430 urn:oasis:names:tc:SAML:2.0:status:Responder

1431 The request could not be performed due to an error on the part of the SAML responder or SAML
1432 authority.

1433 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1434 The SAML responder could not process the request because the version of the request message was
1435 incorrect.

1436 The following second-level status codes are referenced at various places in this specification. Additional
1437 second-level status codes MAY be defined in future versions of the SAML specification. SAML system
1438 entities are free to define more specific status codes by defining appropriate URI references.

1439 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1440 The responding provider was unable to successfully authenticate the principal.

1441 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1442 The responding provider does not support the specified name identifier format for the requested
1443 subject.

1444 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1445 The specified authentication context requirements cannot be met by the responder.

1446 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP
 1447 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an
 1448 <IDPList> can be resolved or that none of the supported identity providers are available.

1449 urn:oasis:names:tc:SAML:2.0:status:NoPassive
 1450 Indicates the identity provider cannot authenticate the principal passively, as has been requested.

1451 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
 1452 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
 1453 supported by the intermediary.

1454 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
 1455 Indicates that an identity provider cannot authenticate the principal directly and is not permitted to
 1456 proxy the request further.

1457 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
 1458 The SAML responder or SAML authority is able to process the request but has chosen not to respond.
 1459 This status code MAY be used when there is concern about the security context of the request
 1460 message or the sequence of request messages received from a particular requester.

1461 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
 1462 The SAML responder or SAML authority does not support the request.

1463 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
 1464 The SAML responder cannot process any requests with the protocol version specified in the request.

1465 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
 1466 The SAML responder cannot process the request because the protocol version specified in the
 1467 request message is a major upgrade from the highest protocol version supported by the responder.

1468 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
 1469 The SAML responder cannot process the request because the protocol version specified in the
 1470 request message is too low.

1471 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
 1472 The resource value provided in the request message is invalid or unrecognized.

1473 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
 1474 The response message would contain more elements than the SAML responder is able to return.

1475 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
 1476 The responding provider does not recognize the principal specified or implied by the request.

1477 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
 1478 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
 1479 request.

1480 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
 1481 type:

```

1482 <element name="StatusCode" type="samlp:StatusCodeType"/>
1483 <complexType name="StatusCodeType">
1484   <sequence>
1485     <element ref="samlp:StatusCode" minOccurs="0"/>
1486   </sequence>
1487   <attribute name="Value" type="anyURI" use="required"/>

```

1488 `</complexType>`

1489 **3.2.1.4 Element <StatusMessage>**

1490 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1491 The following schema fragment defines the <StatusMessage> element:

```
1492 <element name="StatusMessage" type="string"/>
```

1493 **3.2.1.5 Element <StatusDetail>**

1494 The <StatusDetail> element MAY be used to specify additional information concerning an error
1495 condition. The additional information consists of zero or more elements from any namespace, with no
1496 requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

1497 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1498 complex type:

```
1499 <element name="StatusDetail" type="sampl:StatusDetailType"/>  
1500 <complexType name="StatusDetailType">  
1501   <sequence>  
1502     <any namespace="##any" processContents="lax" minOccurs="0"  
1503     maxOccurs="unbounded"/>  
1504   </sequence>  
1505 </complexType>
```

1506 **3.3 Assertion Query and Request Protocol**

1507 This section defines messages and processing rules for requesting existing assertions by reference or
1508 querying for assertions by subject and statement type.

1509 **3.3.1 Element <AssertionIDRequest>**

1510 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>
1511 message element can be used to request that they be returned in a <Response> message. The
1512 <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for
1513 more information on this element.

1514 The following schema fragment defines the <AssertionIDRequest> element:

```
1515 <element name="AssertionIDRequest" type="sampl:AssertionIDRequestType"/>  
1516 <complexType name="AssertionIDRequestType">  
1517   <complexContent>  
1518     <extension base="sampl:RequestAbstractType">  
1519       <sequence>  
1520         <element ref="saml:AssertionIDRef"  
1521         maxOccurs="unbounded"/>  
1522       </sequence>  
1523     </extension>  
1524   </complexContent>  
1525 </complexType>
```

1526 **3.3.2 Queries**

1527 The following sections define the SAML query request messages.

1528 3.3.2.1 Element <SubjectQuery>

1529 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1530 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and
1531 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the
1532 <saml:Subject> element to **RequestAbstractType**.

1533 The following schema fragment defines the <SubjectQuery> element and its
1534 **SubjectQueryAbstractType** complex type:

```
1535 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1536 <complexType name="SubjectQueryAbstractType" abstract="true">  
1537   <complexContent>  
1538     <extension base="samlp:RequestAbstractType">  
1539       <sequence>  
1540         <element ref="saml:Subject"/>  
1541       </sequence>  
1542     </extension>  
1543   </complexContent>  
1544 </complexType>
```

1545 3.3.2.2 Element <AuthnQuery>

1546 The <AuthnQuery> message element is used to make the query "What assertions containing
1547 authentication statements are available for this subject?" A successful <Response> will contain one or
1548 more assertions containing authentication statements.

1549 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1550 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1551 that have occurred in a previous interaction between the indicated subject and the authentication authority.

1552 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1553 the following element and attribute:

1554 **SessionIndex** [Optional]

1555 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1556 containing authentication statements do you have for this subject within the context of the supplied
1557 session information?"

1558 <RequestedAuthnContext> [Optional]

1559 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1560 containing authentication statements do you have for this subject that satisfy the authentication
1561 context requirements in this element?"

1562 In response to an authentication query, a SAML authority returns assertions with authentication
1563 statements as follows:

- 1564 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1565 assertions that may be returned.
- 1566 • If the **SessionIndex** attribute is present in the query, at least one <AuthnStatement> element in
1567 the set of returned assertions MUST contain an **SessionIndex** attribute that matches the
1568 **SessionIndex** attribute in the query. It is OPTIONAL for the complete set of all such matching
1569 assertions to be returned in the response.
- 1570 • If the <RequestedAuthnContext> element is present in the query, at least one
1571 <AuthnStatement> element in the set of returned assertions MUST contain an
1572 <AuthnContext> element that satisfies the element in the query (see Section 3.3.2.3). It is
1573 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1574 The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex
1575 type:

```
1576 <element name="AuthnQuery" type="samlp:AuthnQueryType"/>
1577 <complexType name="AuthnQueryType">
1578   <complexContent>
1579     <extension base="samlp:SubjectQueryAbstractType">
1580       <sequence>
1581         <element ref="samlp:RequestedAuthnContext"
1582 minOccurs="0"/>
1583       </sequence>
1584       <attribute name="SessionIndex" type="string"
1585 use="optional"/>
1586     </extension>
1587   </complexContent>
1588 </complexType>
```

1589 3.3.2.3 Element <RequestedAuthnContext>

1590 The <RequestedAuthnContext> element specifies the authentication context requirements of
1591 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
1592 complex type defines the following elements and attributes:

1593 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1594 Specifies one or more URI references identifying authentication context classes or declarations. (For
1595 more information about authentication context classes, see [SAMLAuthnCxt].)

1596 Comparison [Optional]

1597 Specifies the comparison method used to evaluate the requested context classes or statements, one
1598 of "exact", "minimum", "maximum", or "better". The default is "exact".

1599 Either a set of class references or a set of declaration references can be used. The set of supplied
1600 references MUST be evaluated as an ordered set, where the first element is the most preferred
1601 authentication context class or declaration. If none of the specified classes or declarations can be satisfied
1602 in accordance with the rules below, then the responder MUST return a <Response> message with a
1603 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1604 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
1605 statement MUST be the exact match of at least one of the authentication contexts specified.

1606 If Comparison is set to "minimum", then the resulting authentication context in the authentication
1607 statement MUST be at least as strong (as deemed by the responder) as one of the authentication
1608 contexts specified.

1609 If Comparison is set to "better", then the resulting authentication context in the authentication statement
1610 MUST be stronger (as deemed by the responder) than any one of the authentication contexts specified.

1611 If Comparison is set to "maximum", then the resulting authentication context in the authentication
1612 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength
1613 of at least one of the authentication contexts specified.

1614 The following schema fragment defines the <RequestedAuthnContext> element and its
1615 **RequestedAuthnContextType** complex type:

```
1616 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1617 <complexType name="RequestedAuthnContextType">
1618   <choice>
1619     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1620     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1621   </choice>
```

```

1622     <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1623 use="optional"/>
1624 </complexType>
1625 <simpleType name="AuthnContextComparisonType">
1626     <restriction base="string">
1627         <enumeration value="exact"/>
1628         <enumeration value="minimum"/>
1629         <enumeration value="maximum"/>
1630         <enumeration value="better"/>
1631     </restriction>
1632 </simpleType>

```

1633 3.3.2.4 Element <AttributeQuery>

1634 The <AttributeQuery> element is used to make the query “Return the requested attributes for this
1635 subject.” A successful response will be in the form of assertions containing attribute statements, to the
1636 extent allowed by policy. This element is of type **AttributeQueryType**, which extends
1637 **SubjectQueryAbstractType** with the addition of the following element and attribute:

1638 <saml:AttributeDesignator> [Any Number]

1639 Each <saml:AttributeDesignator> element specifies an attribute whose value is to be returned.
1640 If no attributes are specified, it indicates that all attributes allowed by policy are requested.

1641 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1642 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1643 assertions that may be returned.
- 1644 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute
1645 values returned, as noted above.
- 1646 • The attribute values returned MAY be constrained by application-specific policy considerations.

1647 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1648 complex type:

```

1649 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1650 <complexType name="AttributeQueryType">
1651     <complexContent>
1652         <extension base="samlp:SubjectQueryAbstractType">
1653             <sequence>
1654                 <element ref="saml:AttributeDesignator"
1655 minOccurs="0" maxOccurs="unbounded"/>
1656             </sequence>
1657         </extension>
1658     </complexContent>
1659 </complexType>

```

1660 3.3.2.5 Element <AuthzDecisionQuery>

1661 The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource
1662 be allowed for this subject, given this evidence?” A successful response will be in the form of assertions
1663 containing authorization decision statements.

1664 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no
1665 future enhancements planned. Users who require additional functionality may want to
1666 consider the eXtensible Access Control Markup Language [XACML], which offers
1667 enhanced authorization decision features.

1668 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1669 addition of the following elements and attribute:

1670 Resource [Required]

1671 A URI reference indicating the resource for which authorization is requested.

1672 <saml:Action> [One or More]

1673 The actions for which authorization is requested.

1674 <saml:Evidence> [Optional]

1675 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1676 In response to an authorization decision query, a SAML authority returns assertions with authorization
1677 decision statements as follows:

- 1678 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the
1679 assertions that may be returned.

1680 The following schema fragment defines the <AuthzDecisionQuery> element and its
1681 **AuthzDecisionQueryType** complex type:

```
1682 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>  
1683 <complexType name="AuthzDecisionQueryType">  
1684   <complexContent>  
1685     <extension base="samlp:SubjectQueryAbstractType">  
1686       <sequence>  
1687         <element ref="saml:Action" maxOccurs="unbounded"/>  
1688         <element ref="saml:Evidence" minOccurs="0"/>  
1689       </sequence>  
1690       <attribute name="Resource" type="anyURI" use="required"/>  
1691     </extension>  
1692   </complexContent>  
1693 </complexType>
```

1694 3.3.3 Element <Response>

1695 The <Response> message element is used when a response consists of a list of zero or more assertions
1696 that answer the request. It has the complex type **ResponseType**, which extends **StatusResponseType**
1697 and adds the following elements:

1698 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1699 Specifies an assertion by value, or optionally an encrypted assertion by value. (See Section 2.3.3 for
1700 more information.)

1701 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1702 <element name="Response" type="samlp:ResponseType"/>  
1703 <complexType name="ResponseType">  
1704   <complexContent>  
1705     <extension base="samlp:StatusResponseType">  
1706       <choice minOccurs="0" maxOccurs="unbounded">  
1707         <element ref="saml:Assertion"/>  
1708         <element ref="saml:EncryptedAssertion"/>  
1709       </choice>  
1710     </extension>  
1711   </complexContent>  
1712 </complexType>
```

1713 3.3.4 Processing Rules

1714 In response to a query message, every assertion returned by a SAML authority MUST contain a
1715 `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the query.

1716 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both
1717 apply:

- 1718 • If S2 includes an identifier element (any element whose type is derived from **BaseIDAbstractType**),
1719 then S1 MUST include an identical identifier element, but the element MAY be encrypted (or not) in
1720 either S1 or S2. In other words, the decrypted form of the identifier MUST be identical in S1 and S2.
1721 "Identical" means that the identifier element's content and attribute values MUST be the same. An
1722 encrypted identifier will be identical to the original according to this definition, once decrypted.
- 1723 • If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 MUST include at
1724 least one `<saml:SubjectConfirmation>` element such that the assertion's subject can be
1725 confirmed in the manner described by at least one element in the requested set.

1726 As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular
1727 `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting
1728 S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with
1729 different `Format` values and element content, even if the two identifiers are considered to refer to the
1730 same principal.

1731 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
1732 expressed by a query or assertion reference, the `<Response>` element MUST NOT contain an
1733 `<Assertion>` element and MUST include a `<StatusCode>` element with the value
1734 `urn:oasis:names:tc:SAML:2.0:status:Success`.

1735 All other processing rules associated with the underlying request and response messages MUST be
1736 observed.

1737 3.4 Authentication Request Protocol

1738 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
1739 authentication statements to establish a security context at one or more relying parties, it can use the
1740 authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and
1741 request that it return a `<Response>` message containing one or more such assertions. Such assertions
1742 MAY contain additional statements of any type, but at least one assertion MUST contain at least one
1743 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

1744 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
1745 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
1746 are not specified, though the means of authentication might impact the content of the response. Other
1747 issues related to the validation of authentication credentials by the identity provider or any communication
1748 between the identity provider and any other entities involved in the authentication process are also out of
1749 scope of this protocol.

1750 The descriptions and processing rules in the following sections reference the following actors, many of
1751 whom might be the same entity in a particular profile of use:

1752 Request Issuer

1753 The entity who creates the authentication request and to whom the response is to be returned.

1754 Presenter

1755 The entity who presents the request to the authority and either authenticates itself during the
1756 sending of the message, or relies on an existing security context to establish its identity. If not the

1757 request issuer, the sender acts as an intermediary between the request issuer and the responding
1758 identity provider.

1759 Requested Subject

1760 The entity about whom one or more assertions are being requested.

1761 Confirming Subject

1762 The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>
1763 elements of the resulting assertion(s).

1764 Relying Party

1765 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
1766 the profile or context of use, generally to establish a security context.

1767 3.4.1 Element <AuthnRequest>

1768 To request that an identity provider issue an assertion with an authentication statement, a request issuer
1769 or presenter authenticates to it (or relies on an existing security context) and sends it an
1770 <AuthnRequest> message that describes the properties that the resulting assertion needs to have to
1771 satisfy its purpose. Among these properties may be information that relates to the content of the assertion
1772 and/or information that relates to how the resulting <Response> message should be delivered to the
1773 request issuer.

1774 The request issuer might not be the same as the presenter of the request, if for example the request
1775 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the
1776 requested subject to provide a service.

1777 The <AuthnRequest> message SHOULD be signed or otherwise authenticated and integrity protected
1778 by the protocol binding used to deliver the message.

1779 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
1780 adds the following elements and attributes, all of which are optional in general, but may be required by
1781 specific profiles:

1782 <saml:Subject> [Optional]

1783 Specifies the requested subject of the resulting assertion(s). This may include one or more
1784 <saml:SubjectConfirmation> elements to indicate how and/or by whom the resulting assertions
1785 can be confirmed.

1786 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
1787 requested subject. If no <SubjectConfirmation> elements are included, then the presenter is
1788 presumed to be the only confirming entity required and the method is implied by the profile of use
1789 and/or the policies of the identity provider.

1790 <NameIDPolicy> [Optional]

1791 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
1792 then any type of identifier supported by the identity provider for the requested subject can be used,
1793 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

1794 <saml:Conditions> [Optional]

1795 Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the
1796 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
1797 information in this element is used as input to the process of constructing the assertion, rather than as
1798 conditions on the use of the request itself.

1799 <RequestedAuthnContext> [Optional]
1800 Specifies the requirements, if any, that the request issuer places on the authentication context that
1801 applies to the responding provider's authentication of the presenter. See Section 3.3.2.3 for
1802 processing rules regarding this element.

1803 <Scoping> [Optional]
1804 Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as
1805 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
1806 providers by the responder.

1807 IsPassive [Optional]
1808 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of the
1809 user interface from the request issuer and interact with the presenter in a noticeable fashion. If a value
1810 is not provided, the default is "true".

1811 ForceAuthn [Optional]
1812 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
1813 rely on a previous security context. If a value is not provided, the default is "false". However, if both
1814 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
1815 presenter unless the constraints of IsPassive can be met.

1816 ProtocolBinding [Optional]
1817 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>
1818 message. See [SAMLBind] for more information about protocol bindings and URI references defined
1819 for them.

1820 AssertionConsumerServiceIndex [Optional]
1821 Indirectly identifies the location to which the <Response> message should be returned to the request
1822 issuer. It applies only to profiles in which the request issuer is different from the presenter. The identity
1823 provider MUST have a trusted means to map the index value in the attribute to a location associated
1824 with the request issuer. [SAMLMetadata] provides one possible mechanism. If omitted, then the
1825 identity provider MUST return the <Response> message to the default location associated with the
1826 request issuer for the profile of use.

1827 AssertionConsumerServiceURL [Optional]
1828 Specifies by value the location to which the <Response> message MUST be returned to the request
1829 issuer. The responder MUST ensure by some means that the value specified is in fact associated with
1830 the request issuer. [SAMLMetadata] provides one possible mechanism.

1831 AttributeConsumingServiceIndex [Optional]
1832 Indirectly identifies information associated with the request issuer describing the SAML attributes the
1833 request issuer desires or requires be supplied by the identity provider in the <Response> message.
1834 The identity provider MUST have a trusted means to map the index value in the attribute to
1835 information associated with the request issuer. [SAMLMetadata] provides one possible mechanism.
1836 The identity provider MAY use this information to populate one or more
1837 <saml:AttributeStatement> elements in the assertion(s) it returns.

1838 ProviderName [Optional]
1839 Specifies the human-readable name of the request issuer for use by the presenter's user agent or the
1840 identity provider.

1841 See Section 3.4.1.5 for general processing rules regarding this message.

1842 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**
1843 complex type:

```

1844 <element name="AuthnRequest" type="saml:AuthnRequestType"/>
1845 <complexType name="AuthnRequestType">
1846   <complexContent>
1847     <extension base="saml:RequestAbstractType">
1848       <sequence>
1849         <element ref="saml:Subject" minOccurs="0"/>
1850         <element ref="saml:NameIDPolicy" minOccurs="0"/>
1851         <element ref="saml:Conditions" minOccurs="0"/>
1852         <element ref="saml:RequestedAuthnContext"
1853         minOccurs="0"/>
1854         <element ref="saml:Scoping" minOccurs="0"/>
1855       </sequence>
1856       <attribute name="IsPassive" type="boolean"
1857       use="optional"/>
1858       <attribute name="ForceAuthn" type="boolean"
1859       use="optional"/>
1860       <attribute name="ProtocolBinding" type="anyURI"
1861       use="optional"/>
1862       <attribute name="AssertionConsumerServiceIndex"
1863       type="unsignedShort" use="optional"/>
1864       <attribute name="AssertionConsumerServiceURL"
1865       type="anyURI" use="optional"/>
1866       <attribute name="AttributeConsumingServiceIndex"
1867       type="unsignedShort" use="optional"/>
1868       <attribute name="ProviderName" type="string"
1869       use="optional"/>
1870     </extension>
1871   </complexContent>
1872 </complexType>

```

1873 3.4.1.1 Element <NameIDPolicy>

1874 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
 1875 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

1876 Format [Required]

1877 Specifies the URI reference corresponding to a name identifier format defined in this or another
 1878 specification (see Section 8.3 for examples).

1879 SPNameQualifier [Optional]

1880 Used with a Format of `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` or
 1881 `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`, it optionally specifies that a
 1882 federated identifier be returned (or created) in the namespace of a service provider other than the
 1883 request issuer, or in the namespace of an affiliation group of service providers.

1884 AllowCreate [Optional]

1885 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the
 1886 request, to create a new identifier to represent the principal. Defaults to "true". When "false", the
 1887 request issuer constrains the identity provider to only issue an assertion to it if an acceptable identifier
 1888 for the principal has already been established between them.

1889 When this element is used, if the content is not understood by or acceptable to the identity provider, then
 1890 a <Response> message element MUST be returned with an error <Status>, and MAY contain a
 1891 second-level <StatusCode> of
 1892 `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

1893 A Format of `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates that the
 1894 resulting assertion(s) MUST contain <EncryptedID> elements instead of plaintext. The underlying name
 1895 identifier's unencrypted form can be of any type supported by the identity provider for the requested
 1896 subject.

1897 Any `Format` value (or the omission of this element) MAY result in an `<EncryptedID>` in the resulting
1898 assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

1899 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
1900 complex type:

```
1901 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
1902 <complexType name="NameIDPolicyType">
1903   <sequence/>
1904   <attribute name="Format" type="anyURI" use="required"/>
1905   <attribute name="SPNameQualifier" type="string" use="optional"/>
1906   <attribute name="AllowCreate" type="boolean" use="optional"/>
1907 </complexType>
```

1908 **3.4.1.2 Element <Scoping>**

1909 The `<Scoping>` element specifies the identity providers trusted by the request issuer to authenticate the
1910 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
1911 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
1912 elements and attribute:

1913 `ProxyCount` [Optional]

1914 Specifies the number of proxying indirections permissible between the identity provider that receives
1915 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of
1916 zero permits no proxying, while omitting this attribute expresses no such restriction.

1917 `<IDPList>` [Optional]

1918 An advisory list of identity providers and associated information that the request issuer deems
1919 acceptable to respond to the request.

1920 `<RequesterID>` [Zero or More]

1921 Identifies the set of requesting entities on whose behalf the request issuer is acting. Used to
1922 communicate the chain of request issuers when proxying occurs, as described in Section 3.4.1.6. See
1923 Section 8.3.6 for a description of entity identifiers.

1924 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
1925 `<Response>` message with an error `<Status>` and a second-level `<StatusCode>` of
1926 `urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP` or
1927 `urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP` if it cannot contact or does not support
1928 any of the specified identity providers.

1929 The following schema fragment defines the `<Scoping>` element and its **ScopingType** complex type:

```
1930 <element name="Scoping" type="samlp:ScopingType"/>
1931 <complexType name="ScopingType">
1932   <sequence>
1933     <element ref="samlp:IDPList" minOccurs="0"/>
1934     <element ref="samlp:RequesterID" minOccurs="0"
1935   maxOccurs="unbounded"/>
1936   </sequence>
1937   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
1938 </complexType>
1939 <element name="RequesterID" type="anyURI"/>
```

1940 **3.4.1.3 Element <IDPList>**

1941 The `<IDPList>` element specifies the identity providers trusted by the request issuer to authenticate the
1942 presenter. Its **IDPListType** complex type defines the following elements:

1943 <IDPEntry> [One or More]
1944 Information about a single identity provider.
1945 <GetComplete> [Optional]
1946 If the <IDPList> is not complete, using this element specifies a URI reference that resolves to the
1947 complete list.

1948 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
1949 <element name="IDPList" type="samlp:IDPListType"/>  
1950 <complexType name="IDPListType">  
1951 <sequence>  
1952 <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>  
1953 <element ref="samlp:GetComplete" minOccurs="0"/>  
1954 </sequence>  
1955 </complexType>  
1956 <element name="GetComplete" type="anyURI"/>
```

1957 3.4.1.4 Element <IDPEntry>

1958 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to authenticate
1959 the presenter. Its **IDPEntryType** complex type defines the following attributes:

1960 ProviderID [Required]

1961 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

1962 Name [Optional]

1963 A human-readable name for the identity provider.

1964 Loc [Optional]

1965 A URI reference representing the location of a profile-specific endpoint supporting the authentication
1966 request protocol. The binding to be used must be understood from the profile of use.

1967 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
1968 <element name="IDPEntry" type="samlp:IDPEntryType"/>  
1969 <complexType name="IDPEntryType">  
1970 <sequence/>  
1971 <attribute name="ProviderID" type="anyURI" use="required"/>  
1972 <attribute name="Name" type="string" use="optional"/>  
1973 <attribute name="Loc" type="anyURI" use="optional"/>  
1974 </complexType>
```

1975 3.4.1.5 Processing Rules

1976 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
1977 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
1978 kinds of input and output are required or prohibited. The following processing rules apply as invariant
1979 behavior across any profile of this protocol exchange. All other processing rules associated with the
1980 underlying request and response messages MUST also be observed.

1981 The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing
1982 one or more assertions that meet the specifications defined by the request, or with a <Response>
1983 message containing a <Status> describing the error that occurred. The responder MAY conduct
1984 additional message exchanges with the presenter as needed to initiate or complete the authentication
1985 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
1986 the next section, this includes proxying the request by directing the presenter to another identity provider
1987 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to

1988 authenticate the presenter to the original responder, in effect using SAML as the authentication
1989 mechanism.

1990 If the responder is unable to authenticate the presenter or does not recognize the requested subject, it
1991 MUST return a <Response> with an error <Status>, and MAY return a second-level <StatusCode> of
1992 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
1993 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

1994 If the <saml:Subject> element in the request is present, then the resulting assertions'
1995 <saml:Subject> MUST **strongly match** the request <saml:Subject>, as described in Section 3.3.4,
1996 except that the identifier MAY be in a different format if specified by <NameIDPolicy>. In such a case,
1997 the identifier's physical content MAY be different, but it MUST refer to the same principal.

1998 All of the content defined specifically within <AuthnRequest> is optional, although some may be required
1999 by certain profiles. In the absence of any specific content at all, the following behavior is assumed:

- 2000 • The assertion(s) returned MUST contain a <saml:Subject> element that represents the
2001 presenter. The identifier type and format are determined by the identity provider. At least one
2002 statement MUST be a <saml:AuthnStatement> that describes the authentication performed by
2003 the responder or authentication service associated with it.
- 2004 • The request presenter should, to the extent possible, be the only entity able to satisfy the
2005 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation
2006 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- 2007 • The resulting assertion(s) MUST contain a <saml:AudienceRestriction> element
2008 referencing the request issuer as an acceptable relying party. Other audiences MAY be included as
2009 deemed appropriate by the identity provider.

2010 **3.4.1.6 Proxying**

2011 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or
2012 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
2013 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
2014 <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in
2015 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
2016 identity provider.

2017 Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the
2018 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
2019 proxying provider can issue an assertion of its own in response to the original <AuthnRequest>,
2020 completing the overall message exchange. Both the proxying and authenticating identity providers MAY
2021 include constraints on proxying activity in the messages and assertions they issue, as described in
2022 previous sections and below.

2023 The request issuer can influence proxy behavior by including a <Scoping> element where the provider
2024 sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be
2025 proxied by including an ordered <IDPList> of preferred providers.

2026 An identity provider can control secondary use of its assertions by proxying identity providers using a
2027 <ProxyRestriction> element in the assertions it issues.

2028 **3.4.1.6.1 Proxying Processing Rules**

2029 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is
2030 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2031 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

2032 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity
2033 provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of
2034 `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly
2035 authenticate the presenter.

2036 If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying
2037 identity provider MUST include equivalent or stricter forms of all the information included in the original
2038 request (such as authentication context policy). Note, however, that the proxying provider is free to specify
2039 whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

2040 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
2041 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for
2042 example) will be honored by the authenticating provider.

2043 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less
2044 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new
2045 request SHOULD contain a `<ProxyCount>` attribute.

2046 If an `<IDPList>` was specified in the original request, the new request MUST also contain an
2047 `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the
2048 `<IDPList>`, but MUST NOT remove any from the list.

2049 The authentication request and response are processed in normal fashion, in accordance with the rules
2050 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
2051 provider (in the case of SAML by delivering a `<Response>`), the following steps are followed:

- 2052 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
2053 relevant information from the original assertion or non-SAML equivalent.
- 2054 • The new assertion's `<saml:Subject>` MUST contain an identifier that satisfies the original
2055 request issuer's preferences, as defined by its `<NameIDPolicy>` element.
- 2056 • The `<saml:AuthnStatement>` in the new assertion MUST include a `<saml:AuthnContext>`
2057 element containing a `<saml:AuthenticatingAuthority>` element referencing the identity
2058 provider to which the proxying identity provider referred the presenter. If the original assertion
2059 contains `<saml:AuthnContext>` information that includes one or more
2060 `<saml:AuthenticatingAuthority>` elements, those elements SHOULD be included in the
2061 new assertion, with the new element placed after them.
- 2062 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
2063 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
2064 consistent over time across different requests. The value MUST not conflict with values used or
2065 generated by other SAML providers.
- 2066 • Any other `<saml:AuthnContext>` information MAY be copied, translated, or omitted in
2067 accordance with the policies of the proxying identity provider, provided that the original
2068 requirements dictated by the request issuer are met.

2069 If, in the future, the identity provider is asked to authenticate the same presenter for a second request
2070 issuer, and this request is equally or less strict than the original request (as determined by the proxying
2071 identity provider), the identity provider MAY skip the creation of a new `<AuthnRequest>` to the
2072 authenticating identity provider and immediately issue another assertion (assuming the original assertion
2073 or non-SAML equivalent it received is still valid).

2074 **3.5 Artifact Resolution Protocol**

2075 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
2076 transported in a SAML binding by reference instead of by value. Both requests and responses can be

2077 obtained by reference using this specialized protocol. A message sender, instead of binding a message to
2078 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2079 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver
2080 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
2081 protocol to resolve the artifact into the original protocol message. The most common use for this
2082 mechanism is with bindings that cannot easily carry a message because of size constraints.

2083 Depending on the characteristics of the underlying message being passed by reference, the artifact
2084 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2085 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
2086 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
2087 by any party.

2088 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
2089 as if the message so obtained had been sent originally in place of the artifact.

2090 **3.5.1 Element <ArtifactResolve>**

2091 The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an
2092 <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message.
2093 The original transmission of the artifact is governed by the specific protocol binding that is being used; see
2094 [SAMLBind] for more information on the use of artifacts in bindings.

2095 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
2096 protected by the protocol binding used to deliver the message.

2097 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
2098 adds the following element:

2099 <Artifact> [Required]

2100 The artifact value that the requester received and now wishes to translate into the protocol message it
2101 represents. See [SAMLBind] for specific artifact format information.

2102 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**
2103 complex type:

```
2104 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>  
2105 <complexType name="ArtifactResolveType">  
2106   <complexContent>  
2107     <extension base="samlp:RequestAbstractType">  
2108       <sequence>  
2109         <element ref="samlp:Artifact"/>  
2110       </sequence>  
2111     </extension>  
2112   </complexContent>  
2113 </complexType>  
2114 <element name="Artifact" type="string"/>
```

2115 **3.5.2 Element <ArtifactResponse>**

2116 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
2117 message element. This element is of complex type **ArtifactResponseType**, which extends
2118 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2119 message being returned. This wrapped message element can be a request or a response.

2120 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2121 protected by the protocol binding used to deliver the message.

2122 The following schema fragment defines the <ArtifactResponse> element and its
2123 **ArtifactResponseType** complex type:

```
2124 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2125 <complexType name="ArtifactResponseType">
2126   <complexContent>
2127     <extension base="samlp:StatusResponseType">
2128       <sequence>
2129         <any namespace="##any" processContents="lax"
2130 minOccurs="0"/>
2131       </sequence>
2132     </extension>
2133   </complexContent>
2134 </complexType>
```

2135 **3.5.3 Processing Rules**

2136 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2137 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>
2138 element with no embedded message. In both cases, the <Status> element MUST include a
2139 <StatusCode> element with the code value *Success*. A response message with no embedded
2140 message inside it is termed an empty response in the remainder of this section.

2141 The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent
2142 request with the same artifact by any requester results in an empty response as described above.

2143 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY
2144 be intended for consumption by any party that receives it and can respond appropriately. In most other
2145 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2146 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2147 receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the
2148 original intended recipient, then the artifact issuer MUST return an empty response.

2149 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2150 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2151 return it in an <ArtifactResolve> message to the issuer.

2152 Note that the <ArtifactResponse> message's *InResponseTo* attribute MUST contain the value of
2153 the corresponding <ArtifactResolve> message's *ID* attribute, but the embedded protocol message
2154 will contain its own message identifier, and in the case of an embedded response, may contain a different
2155 *InResponseTo* value that corresponds to the original request message to which the embedded message
2156 is responding.

2157 All other processing rules associated with the underlying request and response messages MUST be
2158 observed.

2159 **3.6 Name Identifier Management Protocol**

2160 After establishing a persistent name identifier for a principal, an identity provider wishing to change the
2161 value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name
2162 identifier will no longer be used to refer to the principal, informs service providers of the change by
2163 sending them a <ManageNameIDRequest> message.

2164 A service provider also uses this message to register or change the *SPProvidedID* value to be included
2165 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
2166 identifier between itself and the identity provider.

2167 3.6.1 Element <ManageNameIDRequest>

2168 A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name
2169 identifier or to indicate the termination of the use of a name identifier.

2170 The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity
2171 protected by the protocol binding used to deliver the message.

2172 This message has the complex type **RegisterNameIDRequestType**, which extends
2173 **RequestAbstractType** and adds the following elements:

2174 <saml:NameID> or <saml:EncryptedID> [Required]

2175 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
2176 principal as currently recognized by the identity and service providers prior to this request.

2177 <NewID> or <NewEncryptedID> or <Terminate> [Required]

2178 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2179 requesting provider concerning this principal, or an indication that the use of the old identifier has
2180 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
2181 appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the
2182 identity provider, the new value will appear in subsequent <NameID> elements as the element's
2183 content.

2184 The following schema fragment defines the <ManageNameIDRequest> element and its
2185 **ManageNameIDRequestType** complex type:

```
2186 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2187 <complexType name="ManageNameIDRequestType">
2188   <complexContent>
2189     <extension base="samlp:RequestAbstractType">
2190       <sequence>
2191         <choice>
2192           <element ref="saml:NameID"/>
2193           <element ref="saml:EncryptedID"/>
2194         </choice>
2195         <choice>
2196           <element ref="samlp:NewID"/>
2197           <element ref="samlp:NewEncryptedID"/>
2198           <element ref="samlp:Terminate"/>
2199         </choice>
2200       </sequence>
2201     </extension>
2202   </complexContent>
2203 </complexType>
2204 <element name="NewID" type="string"/>
2205 <element name="NewEncryptedID" type="saml:EncryptedIDType"/>
2206 <element name="Terminate" type="samlp:TerminateType"/>
2207 <complexType name="TerminateType">
2208   <sequence/>
2209 </complexType>
```

2210 3.6.2 Element <ManageNameIDResponse>

2211 The recipient of a <ManageNameIDRequest> message MUST respond with a
2212 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional
2213 content.

2214 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity
2215 protected by the protocol binding used to deliver the message.

2216 The following schema fragment defines the <ManageNameIDResponse> element:

```
2217 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

2218 3.6.3 Processing Rules

2219 If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize,
2220 the responding provider MUST respond with an error <Status> and MAY respond with a second-level
2221 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2222 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the
2223 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of
2224 an identity provider) it will no longer issue assertions to the service provider about the principal. The
2225 receiving provider can perform any maintenance with the knowledge that the relationship represented by
2226 the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for
2227 whom a relationship has been terminated.

2228 If the service provider requests that its identifier be changed by including a <NewID> (or
2229 <NewEncryptedID>) element, the identity provider MUST include the element's content as the
2230 SPProvidedID when subsequently communicating to the service provider regarding this principal.

2231 If the identity provider requests that its identifier be changed by including a <NewID> (or
2232 <NewEncryptedID>) element, the service provider MUST use the element's content as the
2233 <saml:NameID> element content when subsequently communicating with the identity provider regarding
2234 this principal.

2235 In any case, the <saml:NameID> content in the request and its associated SPProvidedID attribute
2236 MUST contain the most recent name identifier information established between the providers for the
2237 principal.

2238 In the case of an identifier with a Format of urn:oasis:names:tc:SAML:2.0:nameid-
2239 format:persistent or urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted, the
2240 NameQualifier attribute MUST contain the unique identifier of the identity provider or be omitted. If the
2241 identifier was established between the identity provider and an affiliation group of which the service
2242 provider is a member, then the SPNameQualifier attribute MUST contain the unique identifier of the
2243 affiliation group. Otherwise, it MUST contain the unique identifier of the service provider or be omitted.

2244 Changes to these identifiers may take a potentially significant amount of time to propagate through the
2245 systems at both the requester and the responder. Implementations might wish to allow each party to
2246 accept either identifier for some period of time following the successful completion of a name identifier
2247 change. Not doing so could result in the inability of the principal to access resources.

2248 All other processing rules associated with the underlying request and response messages MUST be
2249 observed.

2250 3.7 Single Logout Protocol

2251 The single logout protocol provides a message exchange protocol by which all sessions provided by a
2252 particular session authority are near-simultaneously terminated. The single logout protocol is used either
2253 when a principal logs out at a session participant or when the principal logs out directly at the
2254 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
2255 the logout event can be indicated through the Reason attribute.

2256
2257 The principal may have established authenticated sessions with both the session authority and individual
2258 session participants, based on assertions containing authentication statements supplied by the session
2259 authority.
2260

2261 When the principal invokes the single logout process at a session participant, the session participant
2262 MUST send a <LogoutRequest> message to the session authority that provided the assertion
2263 containing the authentication statement related to that session at the session participant.
2264

2265 When either the principal invokes a logout at the session authority, or a session participant sends a logout
2266 request to the session authority specifying that principal, the session authority MUST send a
2267 <LogoutRequest> message to each session participant to which it provided assertions containing
2268 authentication statements under its current session with the principal, with the exception of the session
2269 participant that sent the <LogoutRequest> message to the session authority.

2270 3.7.1 Element <LogoutRequest>

2271 A session participant or session authority sends a <LogoutRequest> message to indicate that a session
2272 has been terminated.

2273 The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected
2274 by the protocol binding used to deliver the message.

2275 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and
2276 adds the following elements and attributes:

2277 NotOnOrAfter [Optional]

2278 The time at which the request expires. The time value is encoded in UTC, as described in Section
2279 1.2.2.

2280 Reason [Optional]

2281 An indication of the reason for the logout, in the form of a URI reference.

2282 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2283 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as
2284 currently recognized by the identity and service providers prior to this request.

2285 <SessionIndex> [Optional]

2286 The identifier that indexes this session at the message recipient.

2287 The following schema fragment defines the <LogoutRequest> element and associated
2288 **LogoutRequestType** complex type:

```
2289 <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
2290 <complexType name="LogoutRequestType">
2291   <complexContent>
2292     <extension base="samlp:RequestAbstractType">
2293       <sequence>
2294         <choice>
2295           <element ref="saml:BaseID"/>
2296           <element ref="saml:NameID"/>
2297           <element ref="saml:EncryptedID"/>
2298         </choice>
2299         <element ref="samlp:SessionIndex" minOccurs="0"
2300 maxOccurs="unbounded"/>
2301       </sequence>
2302       <attribute name="Reason" type="anyURI" minOccurs="0"/>
2303       <attribute name="NotOnOrAfter" type="dateTime"
2304 minOccurs="0"/>
2305     </extension>
2306   </complexContent>
2307 </complexType>
2308 <element name="SessionIndex" type="string"/>
```

2309 **3.7.2 Element <LogoutResponse>**

2310 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of
2311 type **StatusResponseType**, with no additional content specified.

2312 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
2313 protected by the protocol binding used to deliver the message.

2314 The following schema fragment defines the <LogoutResponse> element:

```
2315 <element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

2316 **3.7.3 Processing Rules**

2317 The message sender MAY use the Reason attribute to indicate the reason for sending the
2318 <LogoutRequest>. The following values are defined by this specification for use by all message
2319 senders; other values MAY be agreed on between participants:

2320 urn:oasis:names:tc:SAML:2.0:logout:user

2321 Specifies that the message is being sent because the principal wishes to terminate the indicated
2322 session.

2323 urn:oasis:names:tc:SAML:2.0:logout:admin

2324 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2325 session for that principal.

2326 All other processing rules associated with the underlying request and response messages MUST be
2327 observed.

2328 Additional processing rules are provided in the following sections.

2329 **3.7.3.1 Session Participant Rules**

2330 When a session participant receives a <LogoutRequest> message, the session participant MUST
2331 authenticate the message. If the sender is the authority that provided an assertion containing an
2332 authentication statement linked to the principal's current session, the session participant MUST invalidate
2333 the principal's session(s) referred to by the <saml:BaseID>, <saml:NameID>, or
2334 <saml:EncryptedID> element, and any <SessionIndex> elements supplied in the message. If no
2335 <SessionIndex> elements are supplied, then all sessions associated with the principal MUST be
2336 invalidated.

2337
2338 The session participant MUST apply the logout request message to any assertion that meets the following
2339 conditions, even if the assertion arrives after the logout request:

- 2340 • The <SessionIndex> of one of the assertion's authentication statements matches one specified
2341 in the logout request, or the logout request contains no <SessionIndex> elements.
- 2342 • The assertion would otherwise be valid.
- 2343 • The logout request has not yet expired (determined by examining the NotOnOrAfter attribute on
2344 the message).

2345 **3.7.3.2 Session Authority Rules**

2346 When a session authority receives a <LogoutRequest> message, the session authority MUST
2347 authenticate the sender. If the sender is a session participant to which the session authority provided an

2348 containing an authentication statement for the current session, then the session authority SHOULD do the
2349 following in the specified order:

- 2350 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session
2351 authority proxied the user's authentication, unless the second authority is the originator of the
2352 `<LogoutRequest>`.
- 2353 • Send a `<LogoutRequest>` message to each session participant for which the session authority
2354 provided assertions in the current session, *other than* the originator of a current
2355 `<LogoutRequest>`.
- 2356 • Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`,
2357 or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout
2358 request message.

2359 If an error occurs during this further processing of the logout (for example, other session participants may
2360 not all implement the particular single logout protocol binding used by the requesting session participant),
2361 then the session authority MUST respond to the original requester with a `<LogoutResponse>` message,
2362 indicating the status of the logout request. The value
2363 `urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding` is provided for a second-level
2364 `<StatusCode>`, indicating that a session participant should retry the `<LogoutRequest>` using a
2365 different protocol binding.

2366 Note that a session authority MAY initiate a logout for reasons other than having received a
2367 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 2368 • If some timeout period was agreed out-of-band with an individual session participant, the session
2369 authority MAY send a `<LogoutRequest>` to that individual participant alone.
- 2370 • An agreed global timeout period has been exceeded.
- 2371 • The principal or some other trusted entity has requested logout of the principal directly at the session
2372 authority.
- 2373 • The session authority has determined that the principal's credentials may have been compromised.

2374 When constructing a logout request message, the session authority MUST set the value of the
2375 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message.

2376 In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also
2377 available for use by the session authority only:

2378 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2379 Specifies that the message is being sent because of the global session timeout interval period
2380 being exceeded.

2381 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

2382 Specifies that the message is being sent because a timeout interval period agreed between a
2383 participant and the authority has been exceeded.

2384 **3.8 Name Identifier Mapping Protocol**

2385 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2386 identifier for the same principal in a particular format or federation namespace, it can send a request to
2387 the identity provider using this protocol.

2388 For example, a service provider that wishes to communicate with another service provider with whom it
2389 does not share an identifier for the principal can use an identity provider that shares an identifier for the

2390 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,
2391 with which it can communicate with the second service provider.

2392 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
2393 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

2394 **3.8.1 Element <NameIDMappingRequest>**

2395 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2396 <NameIDMappingRequest> message. This message has the complex type
2397 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2398 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2399 The identifier and associated descriptive data that specify the principal as currently recognized by the
2400 requester and the responder.

2401 <NameIDPolicy> [Required]

2402 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2403 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2404 binding used to deliver the message.

2405 The following schema fragment defines the <NameIDMappingRequest> element and its
2406 **NameIDMappingRequestType** complex type:

```
2407 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2408 <complexType name="NameIDMappingRequestType">
2409   <complexContent>
2410     <extension base="samlp:RequestAbstractType">
2411       <sequence>
2412         <choice>
2413           <element ref="saml:BaseID"/>
2414           <element ref="saml:NameID"/>
2415           <element ref="saml:EncryptedID"/>
2416         </choice>
2417         <element ref="samlp:NameIDPolicy"/>
2418       </sequence>
2419     </extension>
2420   </complexContent>
2421 </complexType>
```

2422 **3.8.2 Element <NameIDMappingResponse>**

2423 The recipient of a <NameIDMappingRequest> message MUST respond with a
2424 <NameIDMappingResponse> message. This message has the complex type
2425 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following element:

2426 <saml:NameID> or <saml:EncryptedID> [Required]

2427 The identifier and associated attributes that specify the principal in the manner requested, usually in
2428 encrypted form.

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

2431 The following schema fragment defines the <NameIDMappingResponse> element and its
2432 **NameIDMappingResponseType** complex type:

```
2433 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2434 <complexType name="NameIDMappingResponseType">
```

```
2435     <complexContent>
2436         <extension base="samlp:StatusResponseType">
2437             <choice>
2438                 <element ref="saml:NameID"/>
2439                 <element ref="saml:EncryptedID"/>
2440             </choice>
2441         </extension>
2442     </complexContent>
2443 </complexType>
```

2444 **3.8.3 Processing Rules**

2445 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2446 <Status> containing a second-level <StatusCode> of
2447 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2448 At the responder's discretion, the
2449 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to
2450 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2451 All other processing rules associated with the underlying request and response messages MUST be
2452 observed.

2453 4 SAML Versioning

2454 The SAML specification set is versioned in two independent ways. Each is discussed in the following
2455 sections, along with processing rules for detecting and handling version differences. Also included are
2456 guidelines on when and why specific version information is expected to change in future revisions of the
2457 specification.

2458 When version information is expressed as both a Major and Minor version, it may be expressed discretely,
2459 or in the form *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number
2460 $Major_A.Minor_A$ if and only if:

2461 $Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B > Minor_A)$

2462 4.1 SAML Specification Set Version

2463 Each release of the SAML specification set will contain a major and minor version designation describing
2464 its relationship to earlier and later versions of the specification set. The version will be expressed in the
2465 content and filenames of published materials, including the specification set documents and XML schema
2466 documents. There are no normative processing rules surrounding specification set versioning, since it
2467 merely encompasses the collective release of normative specification documents which themselves
2468 contain processing rules.

2469 The overall size and scope of changes to the specification set documents will informally dictate whether a
2470 set of changes constitutes a major or minor revision. In general, if the specification set is backwards
2471 compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain
2472 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
2473 revision.

2474 4.1.1 Schema Version

2475 As a non-normative documentation mechanism, any XML schema documents published as part of the
2476 specification set will contain a `version` attribute on the `<xsd:schema>` element whose value is in the
2477 form *Major.Minor*, reflecting the specification set version in which it has been published. Validating
2478 implementations MAY use the attribute as a means of distinguishing which version of a schema is being
2479 used to validate messages, or to support multiple versions of the same logical schema.

2480 4.1.2 SAML Assertion Version

2481 The SAML `<Assertion>` element contains attributes for expressing the major and minor version of the
2482 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to
2483 document the syntax, semantics, and processing rules of the assertions of the same version. That is,
2484 specification set version 1.0 describes assertion version 1.0, and so on.

2485 There is explicitly NO relationship between the assertion version and the target XML namespace specified
2486 for the schema definitions for that assertion version.

2487 The following processing rules apply:

- 2488 • A SAML authority MUST NOT issue any assertion with an overall *Major.Minor* assertion version
2489 number not supported by the authority.
- 2490 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
2491 supported by the relying party.
- 2492 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version
2493 number is higher than the minor assertion version number supported by the relying party. However,

2494 all assertions that share a major assertion version number MUST share the same general
2495 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For
2496 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the
2497 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the
2498 likely effects of schema evolution.)

2499 **4.1.3 SAML Protocol Version**

2500 The various SAML protocols' request and response elements contain attributes for expressing the major
2501 and minor version of the request or response message using a pair of integers. Each version of the SAML
2502 specification set will be construed so as to document the syntax, semantics, and processing rules of the
2503 protocol messages of the same version. That is, specification set version 1.0 describes request and
2504 response version V1.0, and so on.

2505 There is explicitly NO relationship between the protocol version and the target XML namespace specified
2506 for the schema definitions for that protocol version.

2507 The version numbers used in SAML protocol request and response elements will match for any particular
2508 revision of the SAML specification set.

2509 **4.1.3.1 Request Version**

2510 The following processing rules apply to requests:

- 2511 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2512 SAML requester and the SAML responder.
- 2513 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD
2514 assume that the responder supports requests with the highest request version supported by the
2515 requester.
- 2516 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version
2517 number matching a response version number that the requester does not support.
- 2518 • A SAML responder MUST reject any request with a major request version number not supported by
2519 the responder.
- 2520 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2521 higher than the highest supported request version that it supports. However, all requests that share
2522 a major request version number MUST share the same general processing rules and semantics,
2523 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the
2524 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill
2525 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

2526 **4.1.4 Response Version**

2527 The following processing rules apply to responses:

- 2528 • A SAML responder MUST NOT issue a response message with a response version number higher
2529 than the request version number of the corresponding request message.
- 2530 • A SAML responder MUST NOT issue a response message with a major response version number
2531 lower than the major request version number of the corresponding request message except to
2532 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2533 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
2534 top-level `<StatusCode>` value of
2535 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting

2536 one of the following second-level values:
2537 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,
2538 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, **OR**
2539 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

2540 **4.1.5 Permissible Version Combinations**

2541 Assertions of a particular major version appear only in response messages of the same major version, as
2542 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2543 example, a V1.1 assertion **MAY** appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2544 response message, if the appropriate assertion schema is referenced during namespace importation. But
2545 a V1.0 assertion **MUST NOT** appear in a V2.0 response message because they are of different major
2546 versions.

2547 **4.2 SAML Namespace Version**

2548 XML schema documents published as part of the specification set contain one or more target
2549 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct
2550 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that
2551 part of the specification.

2552 The namespace URI references defined by the specification set will generally contain version information
2553 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI **MUST** correspond
2554 to the major and minor version of the specification set in which the namespace is first introduced and
2555 defined. This information is not typically consumed by an XML processor, which treats the namespace
2556 opaquely, but is intended to communicate the relationship between the specification set and the
2557 namespaces it defines. (This pattern is also followed by the SAML-defined URI-based identifiers that are
2558 listed in Section 8.)

2559 As a general rule, implementers can expect the namespaces (and the associated schema definitions)
2560 defined by a major revision of the specification set to remain valid and stable across minor revisions of the
2561 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but
2562 this is expected to be rare. In such cases, the older namespaces and their associated definitions should
2563 be expected to remain valid until a major specification set revision.

2564 **4.2.1 Schema Evolution**

2565 In general, maintaining namespace stability while adding or changing the content of a schema are
2566 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2567 older implementations will react to any given change, making forward compatibility difficult to achieve.
2568 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
2569 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),
2570 implementations should expect forward-compatible schema changes in minor revisions, allowing new
2571 messages to validate against older schemas.

2572 Implementations **SHOULD** expect and be prepared to deal with new extensions and message types in
2573 accordance with the processing rules laid out for those types. Minor revisions **MAY** introduce new types
2574 that leverage the extension facilities described in Section 7. Older implementations **SHOULD** reject such
2575 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
2576 include new query, statement, or condition types.

2577

5 SAML and XML Signature Syntax and Processing

2578 SAML assertions and SAML protocol request and response messages may be signed, with the following
2579 benefits. An assertion signed by the SAML authority supports assertion integrity, authentication of the
2580 SAML authority to a SAML relying party, and, if the signature is based on the SAML authority's public-
2581 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the
2582 message originator supports message integrity, authentication of message origin to a destination, and, if
2583 the signature is based on the originator's public-private key pair, non-repudiation of origin.

2584 A digital signature is not always required in SAML. For example, in some circumstances, signatures may
2585 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing
2586 protocol response message. "Inherited" signatures should be used with care when the contained object
2587 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context
2588 must be retained to allow validation, exposing the XML content and adding potentially unnecessary
2589 overhead. As another example, the SAML relying party or SAML requester may have obtained an
2590 assertion or protocol message from the SAML authority or SAML responder directly (with no
2591 intermediaries) through a secure channel, with the SAML authority or SAML responder having
2592 authenticated to the relying party or SAML responder by some means other than a digital signature.

2593 Many different techniques are available for "direct" authentication and secure channel establishment
2594 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, and so on. In
2595 addition, the applicable security requirements depend on the communicating applications and the nature
2596 of the assertion or message transported. It is RECOMMENDED that, in all other contexts, digital
2597 signatures be used for assertions and request and response messages. Specifically:

2598 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority
2599 SHOULD be signed by the SAML authority.

2600 • A SAML protocol message arriving at a destination from an entity other than the originating sender
2601 SHOULD be signed by the sender.

2602 Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that
2603 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are
2604 intended to be the primary SAML signature mechanism, but this specification attempts to ensure
2605 compatibility with profiles that may require other mechanisms.

2606 Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be
2607 enveloped.

2608 5.1 Signing Assertions

2609 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as
2610 described in Section 2.

2611 5.2 Request/Response Signing

2612 All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected
2613 in the schema as described in Section 3.

2614 5.3 Signature Inheritance

2615 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
2616 or a request or response, which may be signed. When a SAML assertion does not contain a
2617 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
2618 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,

2619 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
2620 interpretation should be equivalent to the case where the assertion itself was signed with the same key
2621 and signature options.

2622 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
2623 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY
2624 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
2625 information from the surrounding context, but no such inheritance should be inferred unless specifically
2626 identified by the profile.

2627 **5.4 XML Signature Profile**

2628 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
2629 and many choices. This section details constraints on these facilities so that SAML processors do not
2630 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
2631 **xsd:ID**-typed attributes optionally present on the root elements to which signatures can apply, specifically
2632 the **ID** attribute on `<Assertion>` and the various request and response elements. These attributes are
2633 collectively referred to in this section as the identifier attributes.

2634 **5.4.1 Signing Formats and Algorithms**

2635 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
2636 detached.

2637 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
2638 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
2639 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

2640 **5.4.2 References**

2641 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the
2642 enclosing root element. The assertion's or protocol message's root element may or may not be the root
2643 element of the actual XML document containing the signed assertion or protocol message.

2644 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
2645 value of the root element of the message being signed. For example, if the attribute value is "foo", then
2646 the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

2647 **5.4.3 Canonicalization Method**

2648 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
2649 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a
2650 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over
2651 SAML messages embedded in an XML context can be verified independent of that context.

2652 **5.4.4 Transforms**

2653 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
2654 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive
2655 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
2656 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2657 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
2658 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can

2659 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
2660 applying the transforms manually to the content and reverifying the result as consisting of the same SAML
2661 message.

2662 **5.4.5 KeyInfo**

2663 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of
2664 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be
2665 absent.

2666 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

2667 Use of signing does not affect the semantics of statements within assertions in any way, as stated in
2668 Section 2.

2669 **5.4.7 Example**

2670 Following is an example of a signed response containing a signed assertion. Line breaks have been
2671 added for readability; the signatures are not valid and cannot be successfully verified.

2672 TODO: Update example.

```
2673 <Response  
2674   IssueInstant="2003-04-17T00:46:02Z"  
2675   MajorVersion="2"  
2676   MinorVersion="0"  
2677   Recipient="www.opensaml.org"  
2678   ID="_c7055387-af61-4fce-8b98-e2927324b306"  
2679   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"  
2680   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
2681   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2682   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
2683   <ds:Signature  
2684     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
2685     <ds:SignedInfo>  
2686     <ds:CanonicalizationMethod  
2687       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2688     <ds:SignatureMethod  
2689       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
2690     <ds:Reference  
2691       URI="#_c7055387-af61-4fce-8b98-e2927324b306">  
2692     <ds:Transforms>  
2693     <ds:Transform  
2694       Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />  
2695     <ds:Transform  
2696       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">  
2697     <InclusiveNamespaces  
2698       PrefixList="#default saml samlp ds xsd xsi"  
2699       xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2700     </ds:Transform>  
2701     </ds:Transforms>  
2702     <ds:DigestMethod  
2703       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
2704     <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>  
2705     </ds:Reference>  
2706     </ds:SignedInfo>  
2707     <ds:SignatureValue>  
2708     x/GyPbzmfEe85pGD3claXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYwKf+5  
2709     EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D  
2710     w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>  
2711   </ds:KeyInfo>
```

```

2712 <ds:X509Data>
2713 <ds:X509Certificate>
2714 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaksCzAJBgNVBAYTA1VT
2715 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2716 F1VuaXZlcnNpdHkqb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
2717 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
2718 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
2719 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2720 Ym9yMQ4wDAYDVQQKEwVWQ0FJRDEcMBoGAlUEAxMTc2hpYjEuaW50ZXJ2ZXQyLmVh
2721 dTENMCUGCSqGSIB3DQEJARYYcm9vdEBzaGlMS5pbnRlcm5ldDIuZWWR1MIGfMA0G
2722 CSqGSIB3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
2723 IHRyQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIA0APSZB113R6+KYiE7x4XAWIRcP+
2724 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
2725 pmqOIFGTWQIDAQABox0wGzAMBGNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
2726 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2727 qgi7lFV6MDkhmTvtqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
2728 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
2729 </ds:X509Certificate>
2730 </ds:X509Data>
2731 </ds:KeyInfo>
2732 </ds:Signature>
2733 <Status><StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/></Status>
2734 <Assertion
2735   ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2736   IssueInstant="2003-04-17T00:46:02Z"
2737   Issuer="www.opensaml.org"
2738   MajorVersion="2"
2739   MinorVersion="0"
2740   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
2741   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2742   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2743 <Subject>
2744 <NameIdentifier
2745   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
2746 scott@example.org</NameIdentifier>
2747 <SubjectConfirmation>
2748 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
2749 </SubjectConfirmation></Subject>
2750 <SubjectLocality
2751   IPAddress="127.0.0.1"/>
2752 <Conditions
2753   NotBefore="2003-04-17T00:46:02Z"
2754   NotOnOrAfter="2003-04-17T00:51:02Z">
2755 <AudienceRestriction><Audience>http://www.opensaml.org</Audience>
2756 </AudienceRestriction></Conditions>
2757 <AuthnStatement
2758   AuthnInstant="2003-04-17T00:46:00Z"
2759   AuthnMethod="urn:oasis:names:tc:SAML:1.0:am:password">
2760 </AuthnStatement>
2761 <ds:Signature
2762   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2763 <ds:SignedInfo>
2764 <ds:CanonicalizationMethod
2765   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2766 <ds:SignatureMethod
2767   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2768 <ds:Reference
2769   URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
2770 <ds:Transforms>
2771 <ds:Transform
2772   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
2773 <ds:Transform
2774   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2775 <InclusiveNamespaces
2776   PrefixList="#default saml samlp ds xsd xsi"
2777   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />

```

```
2778 </ds:Transform>
2779 </ds:Transforms>
2780 <ds:DigestMethod
2781   Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
2782 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
2783 </ds:Reference>
2784 </ds:SignedInfo>
2785 <ds:SignatureValue>
2786 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
2787 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtp3TD
2788 MWuL/cBUj2OtBZOQMF7jQ9YB7klIz3RqVL+wNmeWI4=</ds:SignatureValue>
2789 <ds:KeyInfo>
2790 <ds:X509Data>
2791 <ds:X509Certificate>
2792 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgakkxCzAJBgNVBAYTA1VT
2793 MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2794 FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lubiBvZiBJ
2795 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
2796 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
2797 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2798 Ym9yMQ4wDAYDVQQKEwVvV0FJRDEcMBoGAlUEAxMTc2hpYjEuaW50ZXJ2ZXQyLmVh
2799 dTEuMCUGCSqGSIb3DQEJARYYcm9vdEBzaGlMS5pbmR1cm5ldDIuZWWR1MIGfMA0G
2800 CSqGSIb3DQEBAAUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
2801 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIaOAPSZB113R6+KYiE7x4XAWIrCP+
2802 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
2803 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMASGA1UdDwQEAwIFoDANBgkq
2804 hkiG9w0BAQQFAAOBqQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2805 qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
2806 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
2807 </ds:X509Certificate>
2808 </ds:X509Data>
2809 </ds:KeyInfo>
2810 </ds:Signature></Assertion></Response>
```

2811

6 SAML and XML Encryption Syntax and Processing

2812 Encryption is used as the means to implement confidentiality. The most common motives for
2813 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for
2814 competitive advantage or similar reasons. Confidentiality may also be required to insure the effectiveness
2815 of some other security mechanism. For example, a secret password or key may be encrypted.

2816 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 2817 • Communications confidentiality may be provided by mechanisms associated with a particular
2818 binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS or
2819 SOAP Message Security mechanisms for confidentiality.
- 2820 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`
2821 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be
2822 encrypted.
- 2823 • An entire assertion may be encrypted, as described in Section 2.3.4.
- 2824 • The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.3.
- 2825 • An `<Attribute>` element may be encrypted, as described in Section 2.4.3.3.

6.1 General Considerations

2827 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use
2828 of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace
2829 the cleartext information in the same location within the XML instance. The `<EncryptedData>` element's
2830 `Type` attribute SHOULD be used and, if it is present, MUST have the value
2831 `http://www.w3.org/2001/04/xmlenc#Element`.

2832 Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The
2833 SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

6.2 Combining Signatures and Encryption

2835 Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and
2836 encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in
2837 the reverse order that signing and encryption were performed.

- 2838 • When the entire assertion is encrypted, the signature MUST first be calculated and in place, and
2839 then the element encrypted.
- 2840 • When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be
2841 performed first and then the signature calculated over the assertion or message containing the
2842 encrypted element.

6.3 Examples

2844 The following example shows an encrypted assertion in a response message:

2845 TBD

2846 The following example shows an encrypted name identifier.

2847 TBD

2848 7 SAML Extensibility

2849 SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas.
2850 An example of an application that extends SAML assertions is the Liberty Protocols and Schema
2851 Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions
2852 and protocols.

2853 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles of use,
2854 which can be combined with extensions to put the SAML framework to new uses.

2855 7.1 Schema Extension

2856 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
2857 elements can serve as the head element of a substitution group. However, SAML types are not defined as
2858 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only
2859 elements and types that have been specifically designed to support extensibility.

2860 7.1.1 Assertion Schema Extension

2861 The SAML assertion schema is designed to permit separate processing of the assertion package and the
2862 statements it contains, if the extension mechanism is used for either part.

2863 The following elements are intended specifically for use as extension points in an extension schema; their
2864 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 2865 • `<BaseID>` and **BaseIDAbstractType**
- 2866 • `<Condition>` and **ConditionAbstractType**
- 2867 • `<Statement>` and **StatementAbstractType**

2868 The following constructs that are directly usable as part of SAML are particularly interesting targets for
2869 extension:

- 2870 • `<AuthnStatement>` and **AuthnStatementType**
- 2871 • `<AttributeStatement>` and **AttributeStatementType**
- 2872 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
- 2873 • `<AudienceRestriction>` and **AudienceRestrictionType**
- 2874 • `<ProxyRestriction>` and **ProxyRestrictionType**
- 2875 • `<OneTimeUse>` and **OneTimeUseType**

2876 7.1.2 Protocol Schema Extension

2877 The following SAML protocol elements are intended specifically for use as extension points in an
2878 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
2879 type:

- 2880 • `<Request>` and **RequestAbstractType**
- 2881 • `<SubjectQuery>` and **SubjectQueryAbstractType**

2882 The following constructs that are directly usable as part of SAML are particularly interesting targets for
2883 extension:

- 2884 • <AuthnQuery> and **AuthnQueryType**
- 2885 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**
- 2886 • <AttributeQuery> and **AttributeQueryType**

2887 7.2 Schema Wildcard Extension Points

2888 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes
2889 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension
2890 schema.

2891 7.2.1 Assertion Extension Points

2892 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- 2893 • <SubjectConfirmationData>: Uses **xsd:anyType**, which allows any sub-elements and
2894 attributes.
- 2895 • <AuthnContextDecl>: Uses **xsd:anyType**, which allows any sub-elements and attributes.
- 2896 • <AttributeValue>: Uses **xsd:anyType**, which allows any sub-elements and attributes.
- 2897 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other
2898 namespaces with lax schema validation processing.

2899 The following constructs in the assertion schema allow arbitrary global attributes:

- 2900 • <AttributeDesignator> and **AttributeDesignatorType**
- 2901 • <Attribute> and **AttributeType** (based on **AttributeDesignatorType**)

2902 7.2.2 Protocol Extension Points

2903 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- 2904 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema
2905 validation processing.
- 2906 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax
2907 schema validation processing.
- 2908 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with
2909 lax schema validation processing. (It is specifically intended to carry a SAML request or response
2910 message element, however.)

2911 7.3 Identifier Extension

2912 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
2913 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
2914 However, it is always possible to define additional URI-based identifiers for these purposes. It is
2915 RECOMMENDED that these additional identifiers be defined in a formal profile of use.

2916

8 SAML-Defined Identifiers

2917 The following sections define URI-based identifiers for common resource access actions, subject name
2918 identifier formats, and attribute name formats.

2919 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of
2920 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
2921 have one of the following stems, according to the specification set version in which they were first
2922 introduced:

```
2923 urn:oasis:names:tc:SAML:1.0:  
2924 urn:oasis:names:tc:SAML:1.1:  
2925 urn:oasis:names:tc:SAML:2.0:
```

8.1 Action Namespace Identifiers

2927 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to
2928 common sets of actions to perform on resources.

8.1.1 Read/Write/Execute/Delete/Control

2930 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

2931 Defined actions:

2932 `Read Write Execute Delete Control`

2933 These actions are interpreted as follows:

2934 `Read`

2935 The subject may read the resource.

2936 `Write`

2937 The subject may modify the resource.

2938 `Execute`

2939 The subject may execute the resource.

2940 `Delete`

2941 The subject may delete the resource.

2942 `Control`

2943 The subject may specify the access control policy for the resource.

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

2945 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

2946 Defined actions:

2947 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

2948 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions
2949 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
2950 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is
2951 affirmatively denied read permission.

2952 A SAML authority MUST NOT authorize both an action and its negated form.

2953 **8.1.3 Get/Head/Put/Post**

2954 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

2955 Defined actions:

2956 GET HEAD PUT POST

2957 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
2958 the GET action on a resource is authorized to retrieve it.

2959 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST
2960 actions to the write permission. The correspondence is not exact however since an HTTP GET operation
2961 may cause data to be modified and a POST operation may cause modification to a resource other than
2962 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

2963 **8.1.4 UNIX File Permissions**

2964 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

2965 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

2966 The action string is a four-digit numeric code:

2967 *extended user group world*

2968 Where the *extended* access permission has the value

2969 +2 if sgid is set

2970 +4 if suid is set

2971 The *user group* and *world* access permissions have the value

2972 +1 if execute permission is granted

2973 +2 if write permission is granted

2974 +4 if read permission is granted

2975 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
2976 and execute; and world read.

2977 **8.2 Attribute Name Format Identifiers**

2978 The following identifiers MAY be used in the NameFormat attribute defined on the
2979 **AttributeDesignatorType** complex type to refer to the classification of the attribute name for purposes of
2980 interpreting the name.

2981 **8.2.1 Unspecified**

2982 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

2983 The interpretation of the attribute name is left to individual implementations.

2984 **8.2.2 URI Reference**

2985 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

2986 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML
2987 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-
2988 specific. See [SAMLProf] for attribute profiles that make use of this identifier.

2989 **8.2.3 Basic**

2990 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

2991 The class of strings acceptable as the attribute name **MUST** be drawn from the set of values belonging to
2992 the primitive type **xsd:Name** as defined in [Schema2] §3.3.6 . See [SAMLProf] for attribute profiles that
2993 make use of this identifier.

2994 **8.3 Name Identifier Format Identifiers**

2995 The following identifiers **MAY** be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
2996 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the
2997 associated processing rules, if any.

2998 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of
2999 SAML.

3000 **8.3.1 Unspecified**

3001 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3002 The interpretation of the content of the element is left to individual implementations.

3003 **8.3.2 Email Address**

3004 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3005 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
3006 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that
3007 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in
3008 parentheses) after it, and is not surrounded by "<" and ">".

3009 **8.3.3 X.509 Subject Name**

3010 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3011 Indicates that the content of the element is in the form specified for the contents of the
3012 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
3013 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
3014 differ from the rules given in IETF RFC 2253 [RFC 2253].

3015 **8.3.4 Windows Domain Qualified Name**

3016 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3017 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3018 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator
3019 MAY be omitted.

3020 **8.3.5 Kerberos Principal Name**

3021 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3022 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3023 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
3024 realm are described in [RFC 1510].

3025 **8.3.6 Entity Identifier**

3026 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3027 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3028 (such as a SAML authority) or is a participant in SAML profiles (such as a service provider supporting the
3029 browser SSO profile). Such an identifier can be used in the <Issuer> element to identify the issuer of a
3030 SAML request, response, or assertion, or within the <NameID> element to make assertions about system
3031 entities that can issue SAML requests, responses, and assertions. It can also be used in other elements
3032 and attributes whose purpose is to identify a system entity in various protocol exchanges.

3033 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3034 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3035 **8.3.7 Persistent Identifier**

3036 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3037 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3038 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
3039 generated by identity providers MUST be constructed using pseudo-random values that have no
3040 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
3041 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
3042 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3043 The element's *NameQualifier* attribute, if present, MUST contain the unique identifier of the identity
3044 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived
3045 from the context of the message containing the element, such as the issuer of an assertion.

3046 The element's *SPNameQualifier* attribute, if present, MUST contain the unique identifier of the service
3047 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be
3048 omitted if the element is contained in a message intended only for consumption directly by the service
3049 provider, and the value would be the name of that service provider.

3050 The element's *SPProvidedID* attribute MUST contain the alternative identifier of the principal most
3051 recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been
3052 established, than the attribute MUST be omitted.

3053 Persistent identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear text
3054 with providers other than the providers that have established the shared identifier. Furthermore, they
3055 MUST NOT appear in log files or similar locations without appropriate controls and protections.
3056 Deployments without such requirements are free to use other kinds of identifiers in their SAML
3057 exchanges, but MUST NOT overload this format with persistent but non-opaque values

3058 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3059 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3060 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3061 different and does not affect the behavior of the identity provider or any processing rules specific to
3062 persistent identifiers in the protocols defined in this specification.

3063 **8.3.8 Transient Identifier**

3064 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

3065 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3066 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3067 accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of
3068 256 characters.

3069 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3070 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
3071 accordance with the rules specified in Section 8.3.7.

3072 **8.4 Consent Identifiers**

3073 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType**
3074 complex type to communicate whether a user gave consent, and under what conditions, for the request.

3075 **8.4.1 Unspecified**

3076 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unspecified

3077 No claim as to user consent is being made.

3078 **8.4.2 Obtained**

3079 **URI:** urn:oasis:names:tc:SAML:2.0:consent:obtained

3080 Indicates that a user's consent has been obtained by the issuer of the request.

3081 **8.4.3 Prior**

3082 **URI:** urn:oasis:names:tc:SAML:2.0:consent:prior

3083 Indicates that a user's consent has been obtained by the issuer of the request at some point prior to the
3084 action that initiated the request.

3085 **8.4.4 Implicit**

3086 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3087 Indicates that a user's consent has been implicitly obtained by the issuer of the request during the action
3088 that initiated the request, as part of a broader indication of consent. Implicit consent is typically more
3089 proximal to the action in time and presentation than prior consent, such as part of a session of activities.

3090 **8.4.5 Explicit**

3091 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3092 Indicates that a user's consent has been explicitly obtained by the issuer of the request during the action
3093 that initiated the request.

3094 **8.4.6 Unavailable**

3095 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3096 Indicates that the issuer of the request did not obtain consent.

3097 **8.4.7 Inapplicable**

3098 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3099 Indicates that the issuer of the request does not believe that they need to obtain or report consent.

9 References

3100

3101 The following works are cited in the body of this specification.

9.1 Normative References

- 3103 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web
3104 Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3105 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
3106 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.
3107 Note that this specification normatively references [Schema2], listed below.
- 3108 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium
3109 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3110 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World
3111 Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3112 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing,
3113 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web
3114 Consortium. Note that this specification normatively references [XMLEnc-XSD],
3115 listed below.
- 3116 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium.
3117 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3118 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January
3119 1999. <http://www.w3.org/TR/REC-xml-names>.
- 3120 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web
3121 Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this
3122 specification normatively references [XMLSig-XSD], listed below.
- 3123 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium.
3124 [http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)
3125 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

9.2 Non-Normative References

- 3127 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty
3128 Alliance Project, January 2003,
3129 [http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf)
3130 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3131 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of*
3132 *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December
3133 1978.
- 3134 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann. *PGP Message Exchange Formats*.
3135 IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 3136 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*
3137 *Certificate and CRL Profile*. IETF RFC 2459, January 1999.
3138 <http://www.ietf.org/rfc/rfc2459.txt>.
- 3139 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF
3140 RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3141 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
3142 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

3143 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
3144 <http://www.ietf.org/rfc/rfc2246.txt>.

3145 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*
3146 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.
3147 <http://www.ietf.org/rfc/rfc2253.txt>.

3148 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF
3149 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3150 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.
3151 <http://www.ietf.org/rfc/rfc2630.txt>.

3152 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.
3153 <http://www.ietf.org/rfc/rfc2822.txt>.

3154 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,
3155 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3156 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF
3157 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3158 [SAMLAuthnCxt] J. Kemp. *Authentication Context for the OASIS Security Assertion Markup*
3159 *Language (SAML)*. OASIS, February 2004. Document ID sstc-saml-authn-
3160 context-2.0. <http://www.oasis-open.org/committees/security/>.

3161 [SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*
3162 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0.
3163 <http://www.oasis-open.org/committees/security/>.

3164 [SAMLProf] E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*
3165 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0.
3166 <http://www.oasis-open.org/committees/security/>.

3167 [SAMLMetadata] E. Maler et al. *Metadata for the OASIS Security Assertion Markup Language*
3168 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-metadata-2.0.
3169 <http://www.oasis-open.org/committees/security/>.

3170 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*
3171 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID
3172 oasis-sstc-saml-conform-2.0. <http://www.oasis-open.org/committees/security/>.

3173 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*
3174 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-
3175 open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3176 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*
3177 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-2.0.
3178 <http://www.oasis-open.org/committees/security/>.

3179 [SAMLXSD] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID
3180 oasis-sstc-saml-schema-protocol-2.0. [http://www.oasis-
3181 open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3182 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*
3183 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID
3184 oasis-sstc-saml-sec-consider-2.0. [http://www.oasis-
3185 open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3186 [SAMLXSD] E. Maler et al. *SAML assertion schema*. OASIS, September 2003. Document ID
3187 oasis-sstc-saml-schema-assertion-2.0. [http://www.oasis-
3188 open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3189 [SAML-TechOvw] J. Hughes et al. *SAML Technical Overview*. OASIS, July 2004. Document ID
3190 oasis-sstc-saml-tech-overview-2.0. [http://www.oasis-
3191 open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3192 **[UNICODE-C]** M. Davis, M. J. Dürst. *Unicode Normalization Forms*. UNICODE Consortium,
3193 March 2001. <http://www.unicode.org/unicode/reports/tr15/tr15-21.html>.

3194 **[W3C-CHAR]** M. J. Dürst. *Requirements for String Identity Matching and String Indexing*. World
3195 Wide Web Consortium, July 1998. <http://www.w3.org/TR/WD-charreq>.

3196 **[W3C-CharMod]** M. J. Dürst. *Character Model for the World Wide Web 1.0: Fundamentals*. World
3197 Wide Web Consortium, February 2004. <http://www.w3.org/TR/charmod/>.

3198 **[X.500]** ITU-T Recommendation X.501: Information Technology - Open Systems
3199 Interconnection - The Directory: Models. 1993.

3200 **[XACML]** eXtensible Access Control Markup Language (XACML), product of the OASIS
3201 XACML TC. [http://www.oasis-](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)
3202 [open.org/committees/tc_home.php?wg_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).

3203 **Appendix A. Acknowledgments**

3204 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
3205 Committee, whose voting members at the time of publication were:

- 3206 • @@

Appendix B. Revision History

Rev	Date	By Whom	What
01	20 Oct 2003	Eve Maler	Initial draft. Converted to OpenOffice. CORE-1 through CORE-4 . Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed. http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf
02	4 Jan 2004	Eve Maler	Implemented Scott Cantor's draft-sstc-nameid-07 solution proposal (http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587) for work item W-2 , Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet). Fixed CORE-10 (the description of subelement occurrence in the <Evidence> element). http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf
03	24 Jan 2004	Scott Cantor	Name identifier, issuer, and federation protocol additions/changes. See 03-interim-diff draft for intermediate set of change bars. http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf
04	1 Feb 2004	Eve Maler	Made minor edits to new and existing material; changed new <AssertionRequest> element name to <AssertionIDRequest>; changed new <AssertionArtifact> and <NewIdentifier> element declarations from local to global; made distinction between normative and non-normative references; implemented the blocking of element substitution. The bulk of work item W-2 , Identity Federation, is now reflected here. What remains is the federation termination protocol, plus a few other pieces that are covered under other work items. http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol (W-2), removed NameID date attributes, clarified Name Reg processing rules, added Extensions facility and Consent attribute. Also moved Signature on assertions to a location consistent with Request and Response. Added session protocol material (W-1); still unfinished. http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf
06	20 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added AssertionURIReference (W-19), a proposal for ProxyRestrictionCondition, and a proposal for AuthNRequest/Response (related to many work items). Fleshed out LogoutRequest/Response (W-1). Implemented the freezing of authZ decision statement functionality (W-28b). http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf

Rev	Date	By Whom	What
07	7 Mar 2004	Scott Cantor, Eve Maler	<p>Implemented new arrangement for subject information and decision on KeyInfo description, as agreed at 2 Mar 2004 telecon.</p> <p>Adjusted normative language around subject "matching" rules based on subject changes.</p> <p>Revised AuthnRequest proposal based on those changes and feedback from list and focus calls.</p> <p>Incorporated additional schema and processing rules related to ECP and proxying use cases from ID-FF.</p> <p>Added AuthnContext to AuthenticationStatement.</p> <p>Added NameIdentifierMapping protocol (W-2).</p>
http://www.oasis-open.org/committees/download.php/5790/ssstc-saml-core-2.0-draft-07-diff.pdf			
08	15 Mar 2004	Scott Cantor, Eve Maler	<p>Added ArtifactRequest/Response pair as a new protocol.</p> <p>Implemented proposed W-28a attribute changes (rev 03 of the proposal, reflecting focus group input).</p>
http://www.oasis-open.org/committees/download.php/5951/ssstc-saml-core-2.0-draft-08-diff.pdf			
09	8 Apr 2004	Eve Maler	<p>Minor cleanup, plus decisions from March-April 2004 F2F meeting: Moved Signature element up in Assertion contents. Clarified that DoNotCacheCondition has one-time-use semantics. Made NameFormat on the Attribute element clearly optional. Changed the default ValueType identifier name. Added the ability to put arbitrary attributes on the AttributeDesignator element. Removed Source on the Attribute element. Changed the content of Extensions in the Request element to ##other. Removed the restriction saying only federated identifiers could be replaced and set with the termination protocol. Changed Reason on the LogoutRequest element to be a URI reference. Made SessionIndex in the LogoutRequest element globally declared. Added bibliographic references to the new SAML specs.</p>
http://www.oasis-open.org/committees/download.php/6323/ssstc-saml-core-2.0-draft-09-diff.pdf			
10	12 Apr 2004	Scott Cantor, Eve Maler	<p>Allowed assertions to be subjectless. Allowed Audience to reference a specific provider URI. Changed AuthnMethod and AuthnContext handling. Removed RelayState. Added AllowCreate on NameIDPolicy. Consolidated two protocols into the name identifier management protocol. Added a name identifier URI for Kerberos principals.</p>
http://www.oasis-open.org/committees/download.php/6347/ssstc-saml-core-2.0-draft-10-diff.pdf			
11	11 May 2004	Eve Maler	<p>Updated the wording describing the permissible combinations of assertion vs. protocol versions (issue TECH-2). Removed the proposed ValueType field on AttributeDesignator; how to do this will be described in the Baseline Attributes spec instead.</p>
http://www.oasis-open.org/committees/download.php/6709/ssstc-saml-core-2.0-draft-11-diff.pdf			
12	17 May 2004	Scott Cantor, Eve Maler	<p>Added ReauthenticateOnOrAfter, shortened various elements and attributes per TC decision, revised text around entity/provider and persistent identifiers, added additional schema and discussion on encryption, turned subject confirmation method into an attribute.</p>

Rev	Date	By Whom	What
http://www.oasis-open.org/committees/download.php/6791/sstc-saml-core-2.0-draft-12-diff.pdf			
13	20 May 2004	Eve Maler, Scott Cantor	Truncated condition subtype names.
http://www.oasis-open.org/committees/download.php/6857/sstc-saml-core-2.0-draft-13-diff.pdf			
14	30 May 2004	Scott Cantor	Various schema enhancements for encryption of NameID, removed AuthnMethod, moved Session attributes, enhanced SubjectConfirmationData with generally useful attributes, added KeyInfoConfirmationDataType, some rewording of persistent NameID format
http://www.oasis-open.org/committees/download.php/6998/sstc-saml-core-2.0-draft-14-diff.pdf			
15	30 June 2004	Scott Cantor	Near final schema cleanup, and feedback from F2F.
http://www.oasis-open.org/committees/download.php/7473/sstc-saml-core-2.0-draft-15-diff.pdf			
16	12 July 2004	Eve Maler, Scott Cantor	Copyediting, clarifications on usage of various elements, addition of Consent values derived from ID-FF 1.2 submission.
http://www.oasis-open.org/committees/download.php/7711/sstc-saml-core-2.0-draft-16-diff.pdf			
17	13 Jul 2004	Eve Maler	Final title-page cleanup for last-call working draft publication.

3208

3209

Appendix C. Notices

3210 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3211 might be claimed to pertain to the implementation or use of the technology described in this document or
3212 the extent to which any license under such rights might or might not be available; neither does it represent
3213 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
3214 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
3215 available for publication and any assurances of licenses to be made available, or the result of an attempt
3216 made to obtain a general license or permission for the use of such proprietary rights by implementors or
3217 users of this specification, can be obtained from the OASIS Executive Director.

3218 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
3219 other proprietary rights which may cover technology that may be required to implement this specification.
3220 Please address the information to the OASIS Executive Director.

3221 **Copyright © OASIS Open 2004. All Rights Reserved.**

3222 This document and translations of it may be copied and furnished to others, and derivative works that
3223 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
3224 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
3225 this paragraph are included on all such copies and derivative works. However, this document itself may
3226 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
3227 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
3228 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
3229 into languages other than English.

3230 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3231 or assigns.

3232 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3233 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3234 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3235 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.