



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

Working Draft 154, 3015 MayJune 2004

Document identifier:

sstc-saml-core-2.0-draft-154

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
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)

38 **Abstract:**

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

41 **Status:**

42 This is a working draft produced by the Security Services Technical Committee. Publication of this
43 draft does not imply TC endorsement. This is an active working draft that may be updated,
44 replaced, or obsoleted at any time. **See the Revision History for details of changes made in**
45 **this revision.**

46 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
47 services@lists.oasis-open.org list. Others should submit them to the [security-services-](mailto:security-services-comment@lists.oasis-open.org)
48 comment@lists.oasis-open.org list (to post, you must subscribe; to subscribe, send a message to
49 security-services-comment-request@lists.oasis-open.org with "subscribe" in the body) or use
50 other OASIS-supported means of submitting comments. The committee will publish vetted errata
51 on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

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

56 Table of Contents

57	1 Introduction.....	6
58	1.1 Notation.....	6
59	1.2 Schema Organization and Namespaces.....	7
60	1.2.1 String and URI Values.....	7
61	1.2.2 Time Values.....	7
62	1.2.3 ID and ID Reference Values.....	7
63	1.2.4 Comparing SAML Values.....	8
64	2 SAML Assertions.....	9
65	2.1 Schema Header and Namespace Declarations.....	9
66	2.2 Simple Types.....	9
67	2.2.1 Simple Type DecisionType.....	10
68	2.3 Name Identifiers.....	10
69	2.3.1 Element <BaseID>.....	10
70	2.3.2 Element <NameID>.....	11
71	2.3.3 Element <EncryptedID>.....	11
72	2.3.4 Element <Issuer>.....	12
73	2.4 Assertions.....	12
74	2.4.1 Element <AssertionIDReference>.....	12
75	2.4.2 Element <AssertionURIReference>.....	13
76	2.4.3 Element <Assertion>.....	13
77	2.4.3.1 Element <Subject>.....	14
78	2.4.3.2 Element <SubjectConfirmation>.....	15
79	2.4.3.3 Element <SubjectConfirmationData>.....	16
80	2.4.3.4 Complex Type KeyInfoConfirmationDataType.....	16
81	2.4.3.5 Element <Conditions>.....	17
82	2.4.3.5.1 Attributes NotBefore and NotOnOrAfter.....	18
83	2.4.3.5.2 Element <Condition>.....	18
84	2.4.3.5.3 Elements <AudienceRestriction> and <Audience>.....	19
85	2.4.3.5.4 Element <OneTimeUse>.....	19
86	2.4.3.5.5 Element <ProxyRestriction>.....	20
87	2.4.3.6 Element <Advice>.....	21
88	2.4.4 Element <EncryptedAssertion>.....	21
89	2.5 Statements.....	22
90	2.5.1 Element <Statement>.....	22
91	2.5.2 Element <AuthnStatement>.....	22
92	2.5.2.1 Element <SubjectLocality>.....	23
93	2.5.2.2 Element <AuthnContext>.....	23
94	2.5.3 Element <AttributeStatement>.....	24
95	2.5.3.1 Elements <AttributeDesignator> and <Attribute>.....	24
96	2.5.3.2 Element <Attribute>.....	25
97	2.5.3.2.1 Element <AttributeValue>.....	26
98	2.5.3.3 Element <EncryptedAttribute>.....	26
99	2.5.4 Element <AuthzDecisionStatement>.....	27
100	2.5.4.1 Element <Action>.....	28
101	2.5.4.2 Element <Evidence>.....	29
102	3 SAML Protocols.....	30
103	3.1 Schema Header and Namespace Declarations.....	30
104	3.2 Requests and Responses.....	31

105	3.2.1 Complex Type RequestAbstractType.....	31
106	3.2.1.1 Complex Type StatusResponseType.....	32
107	3.2.1.2 Element <Status>.....	33
108	3.2.1.3 Element <StatusCode>.....	34
109	3.2.1.4 Element <StatusMessage>.....	36
110	3.2.1.5 Element <StatusDetail>.....	36
111	3.3 Assertion Query and Request Protocol.....	36
112	3.3.1 Element <AssertionIDRequest>.....	36
113	3.3.2 Queries.....	36
114	3.3.2.1 Element <SubjectQuery>.....	37
115	3.3.2.2 Element <AuthnQuery>.....	37
116	3.3.2.3 Element <RequestedAuthnContext>.....	38
117	3.3.2.4 Element <AttributeQuery>.....	39
118	3.3.2.5 Element <AuthzDecisionQuery>.....	39
119	3.3.3 Element <Response>.....	40
120	3.3.4 Processing Rules.....	41
121	3.4 Authentication Request Protocol.....	41
122	3.4.1 Element <AuthnRequest>.....	42
123	3.4.1.1 Element <NameIDPolicy>.....	44
124	3.4.1.3 Element <Scoping>.....	45
125	3.4.1.4 Element <IDPList>.....	46
126	3.4.1.5 Element <IDPEntry>.....	46
127	3.4.1.6 Processing Rules.....	46
128	3.4.1.7 Proxying.....	47
129	3.5 Artifact Resolution Protocol.....	49
130	3.5.1 Element <ArtifactRequestsolve>.....	49
131	3.5.2 Element <ArtifactResponse>.....	50
132	3.5.3 Processing Rules.....	50
133	3.6 Name Identifier Management Protocol.....	51
134	3.6.1 Element <ManageNameIDRequest>.....	51
135	3.6.2 Element <ManageNameIDResponse>.....	52
136	3.6.3 Processing Rules.....	52
137	3.7 Single Logout Protocol.....	53
138	3.7.1 Element <LogoutRequest>.....	53
139	3.7.2 Element <LogoutResponse>.....	54
140	3.7.3 Processing Rules.....	54
141	3.7.3.1 Session Participant Rules.....	55
142	3.7.3.2 Session Authority Rules.....	55
143	3.8 Name Identifier Mapping Protocol.....	56
144	3.8.1 Element <NameIDMappingRequest>.....	56
145	3.8.2 Element <NameIDMappingResponse>.....	57
146	3.8.3 Processing Rules.....	58
147	4 SAML Versioning.....	59
148	4.1 SAML Specification Set Version.....	59
149	4.1.1 Schema Version.....	59
150	4.1.2 SAML Assertion Version.....	59
151	4.1.3 SAML Protocol Version.....	60
152	4.1.3.1 Request Version.....	60
153	4.1.4 Response Version.....	60
154	4.1.5 Permissible Version Combinations.....	61
155	4.2 SAML Namespace Version.....	61
156	4.2.1 Schema Evolution.....	61
157	5 SAML and XML Signature Syntax and Processing.....	62

158	5.1 Signing Assertions.....	63
159	5.2 Request/Response Signing.....	63
160	5.3 Signature Inheritance.....	63
161	5.4 XML Signature Profile.....	63
162	5.4.1 Signing Formats and Algorithms.....	63
163	5.4.2 References.....	63
164	5.4.3 Canonicalization Method.....	64
165	5.4.4 Transforms.....	64
166	5.4.5 KeyInfo.....	64
167	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	64
168	5.4.7 Example.....	64
169	6 SAML and XML Encryption Syntax and Processing.....	67
170	6.1 General Considerations.....	67
171	6.2 Combining Signatures and Encyption.....	67
172	6.3 Examples.....	67
173	7 SAML Extensions.....	68
174	7.1 Assertion Schema Extension.....	68
175	7.2 Protocol Schema Extension.....	69
176	8 SAML-Defined Identifiers.....	70
177	8.1 Action Namespace Identifiers.....	70
178	8.1.1 Read/Write/Execute/Delete/Control.....	70
179	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	70
180	8.1.3 Get/Head/Put/Post.....	71
181	8.1.4 UNIX File Permissions.....	71
182	8.2 Attribute NameFormat Identifiers.....	71
183	8.2.1 Unspecified.....	71
184	8.2.2 URI Reference.....	72
185	8.2.3 Basic.....	72
186	8.3 NameID Format Identifiers.....	72
187	8.3.1 Unspecified.....	72
188	8.3.2 Email Address.....	72
189	8.3.3 X.509 Subject Name.....	72
190	8.3.4 Windows Domain Qualified Name.....	73
191	8.3.5 Kerberos Principal Name.....	73
192	8.3.6 Entity Identifier.....	73
193	8.3.7 Persistent Identifier.....	73
194	8.3.8 Transient Identifier.....	74
195	9 References.....	75
196	9.1 Normative References.....	75
197	9.2 Non-Normative References.....	75
198		

1 Introduction

199

200 This specification defines the syntax and semantics for Security Assertion Markup Language (SAML)
201 assertions and the protocols for requesting and returning them. SAML assertions, requests, and
202 responses are encoded in XML [XML]and use XML namespaces [XMLNS]. They are typically embedded
203 in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The
204 SAML specification for bindings [SAMLBind] provides frameworks for this embedding and transport. Files
205 containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMPL-XSD] are
206 available.

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

1.1 Notation

208

209 This specification uses schema documents conforming to W3C XML Schema and normative text to
210 describe the syntax and semantics of XML-encoded SAML assertions and protocol messages.

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

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

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

219 Listings of SAML schemas appear like this.

220

221 Example code listings appear like this.

222 In cases of disagreement between the SAML schema documents [SAML-XSD] [SAMPL-XSD] and this
223 specification, the schema documents take precedence.

224 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for
225 their respective namespaces (see Section Schema Organization and Namespaces) as follows, whether or
226 not a namespace declaration is present in the example:

- 227 • The prefix `saml:` stands for the SAML assertion namespace,
228 `urn:oasis:names:tc:SAML:2.0:assertion`.
- 229 • The prefix `samlp:` stands for the SAML request-response protocol namespace,
230 `urn:oasis:names:tc:SAML:2.0:protocol`.
- 231 • The prefix `ds:` stands for the W3C XML Signature namespace,
232 <http://www.w3.org/2000/09/xmldsig#> [XMLSig-XSD].
- 233 • The prefix `xenc:` stands for the W3C XML Encryption namespace,
234 <http://www.w3.org/2001/04/xmlenc#> [XMLEnc-XSD].
- 235 • The prefix `xsd:` stands for the W3C XML Schema namespace,
236 <http://www.w3.org/2001/XMLSchema> [Schema1], in example listings. In schema listings, this is
237 the default namespace and no prefix is shown.

238 This specification uses the following typographical conventions in text: `<SAMLElement>`,
239 `<ns:ForeignElement>`, Attribute, **Datatype**, OtherCode.

240 1.2 Schema Organization and Namespaces

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

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

244 The SAML request-response protocol structures are defined in a schema [SAML-XP] associated with
245 the following XML namespace:

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

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

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

250 See Section SAML Namespace Version for information on SAML namespace versioning.

251 1.2.1 String and URI Values

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

258 1.2.2 Time Values

259 All SAML time values have the type **xsd:dateTime**, which is built in to the W3C XML Schema Datatypes
260 specification [Schema1], and MUST be expressed in UTC form.

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

263 1.2.3 ID and ID Reference Values

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

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

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

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

283 1.2.4 Comparing SAML Values

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

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

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

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

2 SAML Assertions

304

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

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

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

2.1 Schema Header and Namespace Declarations

318

319 The following schema fragment defines the XML namespaces and other header information for the
320 assertion schema:

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

2.2 Simple Types

346

347 The following section defines the SAML assertion-related simple types.

348 2.2.1 Simple Type DecisionType

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

351 Permit

352 The specified action is permitted.

353 Deny

354 The specified action is denied.

355 Indeterminate

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

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

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

```
361 <simpleType name="DecisionType">  
362   <restriction base="string">  
363     <enumeration value="Permit"/>  
364     <enumeration value="Deny"/>  
365     <enumeration value="Indeterminate"/>  
366   </restriction>  
367 </simpleType>
```

368 2.3 Name Identifiers

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

371 2.3.1 Element <BaseID>

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

375 NameQualifier [Optional]

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

378 SPNameQualifier [Optional]

379 Further qualifies an identifier with the name of ~~the~~ service provider or affiliation of providers. This
380 attribute provides an additional means to federate identifiers on the basis of the relying party or
381 parties which has federated the principal's identity.

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

```
383 <element name="BaseID" type="saml:BaseIDAbstractType"/>  
384 <complexType name="BaseIDAbstractType" abstract="true" mixed="true">  
385   <complexContent>  
386     <extension base="anyType">  
387       <attribute name="NameQualifier" type="string" use="optional"/>
```

```

388         <attribute name="SPNameQualifier" type="string" use="optional"/>
389     </extension>
390 </complexContent>
391 </complexType>

```

392 2.3.2 Element <NameID>

393 The <NameID> element is of type **NameIDType**, which restricts **BaseIDAbstractType** to simple string
394 content and provides additional attributes as follows:

395 Format [Optional]

396 A URI reference representing the classification of string-based identifier information. See Section
397 NameID Format Identifiers for some URI references that MAY be used as the value of the
398 Format attribute and their associated descriptions and processing rules. If no Format value is
399 provided, the identifier
400 [urn:oasis:names:tc:SAML:1.0:urn:oasis:names:tc:SAML:1.0:nameid-](#)
401 [format:unspecified](#) (see Section Unspecified) is in effect.

402 When a Format value other than those specified in Section NameID Format Identifiers is used,
403 the content of the <NameID> element is to be interpreted according to the specification of that
404 format as defined outside of this specification. If not otherwise indicated by the specification of the
405 format, issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the
406 asserting and relying parties are implementation-specific.

407 SPProvidedID [Optional]

408 [The](#) name identifier established by the service provider or affiliation of providers for the principal,
409 if different from the primary name identifier given in the content of the <NameID> element. [This](#)
410 [attribute provides a means of integrating the use of SAML with existing identifiers already in use](#)
411 [by a service provider.](#)

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

```

413 <element name="NameID" type="saml:NameIDType"/>
414 <complexType name="NameIDType" mixed="false">
415     <simpleContent>
416         <restriction base="saml:BaseIDAbstractType">
417             <simpleType>
418                 <restriction base="string"/>
419             </simpleType>
420             <attribute name="Format" type="anyURI" use="optional"/>
421             <attribute name="SPProvidedID" type="string" use="optional"/>
422         </restriction>
423     </simpleContent>
424 </complexType>

```

425 2.3.3 Element <EncryptedID>

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

429 <xenc:EncryptedData> [Required]

430 The encrypted content and associated encryption details, as defined by the XML Encryption
431 Syntax and Processing specification [XMLEnc]. The Type attribute SHOULD be present and
432 MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted

433 content MUST contain an element that has a type that is derived from **BaseIDAbstractType** or
434 from **AssertionType**.

435 <xenc:EncryptedKey> [Zero or more]

436 Wrapped decryption keys, as defined by **[XMLEnc]**. Each wrapped key SHOULD include a
437 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
438 the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity as defined by
439 section 8.3.6.

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

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

```
445 <element name="EncryptedID" type="saml:EncryptedIDType"/>
446 <complexType name="EncryptedIDType" mixed="false">
447 <complexContent>
448 <restriction base="saml:BaseIDType">
449 <sequence>
450 <element ref="xenc:EncryptedData"/>
451 <element ref="xenc:EncryptedKey" minOccurs="0"
452 maxOccurs="unbounded"/>
453 </sequence>
454 </restriction>
455 </complexContent>
456 </complexType>
```

457 2.3.4 Element <Issuer>

458 The <Issuer> element, with complex type **NameIDType**, provides information about the issuer of a
459 SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name,
460 but permits various attributes of descriptive metadata. ~~The Format attribute typically contains the value if~~
461 ~~no Format value is provided, the identifier~~ `urn:oasis:names:tc:SAML:2.0:nameid-`
462 `format:entity` ~~is in effect~~.

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

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

465 2.4 Assertions

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

467 2.4.1 Element <AssertionIDReference>

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

471 The following schema fragment defines the <AssertionIDReference> element:

```
472 <element name="AssertionIDReference" type="NCName"/>
```

473 | 2.4.2 Element <AssertionURIReference>

474 | The <AssertionURIReference> element makes a reference to a SAML assertion by its uniform
475 | resource identifier (URI). Dereferencing the URI (in a fashion dictated by the URI) is intended to produce
476 | the assertion. See the Bindings specification [SAMLBind] for information on how this element is used in a
477 | protocol binding.

478 | The following schema fragment defines the <AssertionURIReference> element:

```
479 | <element name="AssertionURIReference" type="anyURI"/>
```

480 | 2.4.3 Element <Assertion>

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

483 | MajorVersion [Required]

484 | The major version of this assertion. The identifier for the version of SAML defined in this specification
485 | is 2. SAML versioning is discussed in Section SAML Versioning.

486 | MinorVersion [Required]

487 | The minor version of this assertion. The identifier for the version of SAML defined in this specification
488 | is 0. SAML versioning is discussed in Section SAML Versioning.

489 | ID [Required]

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

492 | IssueInstant [Required]

493 | The time instant of issue in UTC, as described in Section Time Values.

494 | <Issuer> [Required]

495 | The SAML authority that is making the claim(s) in the assertion. The issuer identity SHOULD be
496 | unambiguous to the intended relying parties. ~~If the Format attribute is omitted, the identifier~~
497 | ~~urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see section 8.3.1) is~~
498 | ~~assumed.~~

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

502 | <ds:Signature> [Optional]

503 | An XML Signature that authenticates the assertion, as described in Section SAML and XML Signature
504 | Syntax and Processing.

505 | <Subject> [Optional]

506 | The subject of the statement(s) in the assertion.

507 | <Conditions> [Optional]

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

509 | <Advice> [Optional]

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

512 Zero or more of the following statement elements:

513 <Statement>

514 A statement defined in an extension schema.

515 <AuthnStatement>

516 An authentication statement.

517 <AuthzDecisionStatement>

518 An authorization decision statement.

519 <AttributeStatement>

520 An attribute statement.

521 An assertion with no statements MUST contain a <Subject> element. Such an assertion identifies a principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further information associated with that principal.; a assertion containing no statements binds the name identifier in the subject to a principal.

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

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

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

```
531 <element name="Assertion" type="saml:AssertionType"/>
532 <complexType name="AssertionType">
533   <sequence>
534     <element ref="saml:Issuer"/>
535     <element ref="ds:Signature" minOccurs="0"/>
536     <element ref="saml:Subject" minOccurs="0"/>
537     <element ref="saml:Conditions" minOccurs="0"/>
538     <element ref="saml:Advice" minOccurs="0"/>
539     <choice minOccurs="0" maxOccurs="unbounded">
540       <element ref="saml:Statement"/>
541       <element ref="saml:AuthnStatement"/>
542       <element ref="saml:AuthzDecisionStatement"/>
543       <element ref="saml:AttributeStatement"/>
544     </choice>
545   </sequence>
546   <attribute name="MajorVersion" type="integer" use="required"/>
547   <attribute name="MinorVersion" type="integer" use="required"/>
548   <attribute name="ID" type="ID" use="required"/>
549   <attribute name="IssueInstant" type="dateTime" use="required"/>
550 </complexType>
```

551 2.4.3.1 Element <Subject>

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

555 <NameID>, <EncryptedID>, Or <BaseID>

556 Identifies the subject.

557 <SubjectConfirmation>

558 Information that allows the subject to be authenticated. If more than one subject confirmation is

559 provided, then usage of any one of them is sufficient to confirm the subject for the purpose of applying
560 the assertion.

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

```
567 <element name="Subject" type="saml:SubjectType"/>
568 <complexType name="SubjectType">
569   <choice>
570     <sequence>
571       <choice>
572         <element ref="saml:BaseID"/>
573         <element ref="saml:NameID"/>
574         <element ref="saml:EncryptedID"/>
575       </choice>
576       <element ref="saml:SubjectConfirmation" minOccurs="0"
577 maxOccurs="unbounded"/>
578     </sequence>
579     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
580   </choice>
581 </complexType>
```

582 2.4.3.2 Element <SubjectConfirmation>

583 The <SubjectConfirmation> element ~~specifies a subject by supplying data that allows the subject to~~
584 ~~be authenticated~~provides the means for a relying party to verify the correspondence of the subject of the
585 assertion with the party with whom they are communicating. It contains the following attributes and
586 elements:

587 Method [Required]

588 A URI reference that identifies a protocol to be used to authenticateconfirm the subject. URI
589 references identifying SAML-defined confirmation methods are currently defined with the SAML
590 profiles in the SAML profiles specification [SAMLProf]. Additional methods may be added by defining
591 new URIs and profiles or by private agreement.

592 <SubjectConfirmationData> [Optional]

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

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

```
601 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
602 <complexType name="SubjectConfirmationType">
603   <sequence>
604     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
605   </sequence>
606   <attribute name="Method" type="anyURI"/>
607 </complexType>
```


609 2.4.3.3 Element <SubjectConfirmationData>

610 The <SubjectConfirmationData> element and its **SubjectConfirmationDataType** complex type
611 specifies additional data that allows the subject to be [authenticatedconfirmed](#) or constrains the
612 circumstances under which the [authenticationconfirmation](#) can take place. It contains the following
613 optional attributes that can apply to any method:

614 NotBefore [Optional]

615 | A time instant before which the subject cannot be [authenticatedconfirmed](#).

616 NotOnOrAfter [Optional]

617 | A time instant at which the subject can no longer be [authenticatedconfirmed](#).

618 Recipient [Optional]

619 | Specifies the entity or location to which an entity can present the assertion while
620 | [authenticatingconfirming](#) itself.

621 InResponseTo [Optional]

622 | Specifies the RequestID of a SAML protocol message in response to which an entity can present
623 | the assertion while [authenticatingconfirming](#) itself.

624 | ~~IP~~Address [Optional]

625 | Specifies the [networkIP](#) address from which an entity can present the assertion while authenticating
626 | itself.

627 Particular confirmation methods MAY require the use of one or more of these attributes. Note that the time
628 period specified by the optional NotBefore and NotOnOrAfter attributes, if any, SHOULD fall within the
629 overall assertion validity period as specified by the <Conditions> element's NotBefore and
630 NotOnOrAfter attributes.

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

```
633 <element name="SubjectConfirmationData"  
634 type="saml:SubjectConfirmationDataType"/>  
635 <complexType name="SubjectConfirmationDataType" mixed="true">  
636 <complexContent>  
637 <extension base="anyType">  
638 <attribute name="NotBefore" type="dateTime"  
639 use="optional"/>  
640 <attribute name="NotOnOrAfter" type="dateTime"  
641 use="optional"/>  
642 <attribute name="Recipient" type="anyURI" use="optional"/>  
643 <attribute name="InResponseTo" type="NCName"  
644 use="optional"/>  
645 <attribute name="IPAddress" type="string" use="optional"/>  
646 </extension>  
647 </complexContent>  
648 </complexType>
```

649 2.4.3.4 Complex Type KeyInfoConfirmationDataType

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

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

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

```
658 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
659   <complexContent>  
660     <restriction base="saml:SubjectConfirmationDataType">  
661       <sequence>  
662         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>  
663       </sequence>  
664     </restriction>  
665   </complexContent>  
666 </complexType>
```

667 2.4.3.5 Element `<Conditions>`

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

669 `NotBefore` [Optional]

670 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC as
671 described in Section Time Values.

672 `NotOnOrAfter` [Optional]

673 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC as
674 described in Section Time Values.

675 `<Condition>` [Any Number]

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

677 `<AudienceRestriction>` [Any Number]

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

679 `<OneTimeUse>` [OptionalAny Number]

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

683 `<ProxyRestriction>` [OptionalAny Number]

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

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

```
689 <element name="Conditions" type="saml:ConditionsType"/>  
690 <complexType name="ConditionsType">  
691   <choice minOccurs="0" maxOccurs="unbounded">  
692     <element ref="saml:AudienceRestriction"/>  
693     <element ref="saml:OneTimeUse">  
694     <element ref="saml:ProxyRestriction"/>  
695     <element ref="saml:Condition"/>  
696   </choice>  
697   <attribute name="NotBefore" type="dateTime" use="optional"/>  
698   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
699 </complexType>
```

700 If an assertion contains a `<Conditions>` element, the validity of the assertion is dependent on the sub-
701 elements and attributes provided. When processing the sub-elements and attributes of a `<Conditions>`
702 element, the following rules MUST be used in the order shown to determine the overall validity of the
703 assertion:

- 704 1. If no sub-elements or attributes are supplied in the `<Conditions>` element, then the assertion is
705 considered to be **Valid**.
- 706 2. If any sub-element or attribute of the `<Conditions>` element is determined to be invalid, then the
707 assertion is **Invalid**.
- 708 3. If any sub-element or attribute of the `<Conditions>` element cannot be evaluated, then the validity
709 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 710 4. If all sub-elements and attributes of the `<Conditions>` element are determined to be **Valid**, then the
711 assertion is considered to be **Valid**.

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

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

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

720 **2.4.3.5.1 Attributes NotBefore and NotOnOrAfter**

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

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

726 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
727 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the
728 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the
729 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),
730 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither
731 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is valid
732 at any time.

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

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

737 **2.4.3.5.2 Element <Condition>**

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

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

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

744 2.4.3.5.3 Elements <AudienceRestriction> and <Audience>

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

750 <Audience>

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

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

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

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

```
764 <element name="AudienceRestriction"  
765 type="saml:AudienceRestrictionType"/>  
766 <complexType name="AudienceRestrictionType">  
767 <complexContent>  
768 <extension base="saml:ConditionAbstractType">  
769 <sequence>  
770 <element ref="saml:Audience" maxOccurs="unbounded"/>  
771 </sequence>  
772 </extension>  
773 </complexContent>  
774 </complexType>  
775 <element name="Audience" type="anyURI"/>
```

776 2.4.3.5.4 Element <OneTimeUse>

777 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be
778 retained for future use. Note that no relying party is required to perform caching. However, any that do so
779 MUST observe this condition. This condition conveys one-time-use semantics, and is independent from
780 the NotBefore and NotOnOrAfter condition information.

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

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

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

```
787 <element name="OneTimeUse" type="saml:OneTimeUseType"/>
788 <complexType name="OneTimeUseType">
789   <complexContent>
790     <extension base="saml:ConditionAbstractType"/>
791   </complexContent>
792 </complexType>
```

793 **2.4.3.5.5 Element <ProxyRestriction>**

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

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

799 Count [Optional]

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

802 <Audience> [Zero or More]

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

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

807 If no <Audience> elements are specified, then no restrictions are made upon the relying parties to whom
808 subsequent assertions can be issued. Otherwise, any assertions so issued **MUST** themselves contain an
809 <AudienceRestriction> element with at least one of the <Audience> elements present in the
810 previous <ProxyRestriction> element, and no <Audience> elements present that were not in the
811 previous <ProxyRestriction> element.

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

814 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>
815 is considered to always be valid.

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

```
818 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
819 <complexType name="ProxyRestrictionType">
820   <complexContent>
821     <extension base="saml:ConditionAbstractType">
822       <sequence>
823         <element ref="saml:Audience" minOccurs="0"
824 maxOccurs="unbounded"/>
825       </sequence>
826       <attribute name="Count" type="nonNegativeInteger"
827 use="optional"/>
828     </extension>
829   </complexContent>
830 </complexType>
```

831 2.4.3.6 Element <Advice>

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

835 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,
836 <AssertionIDReference>, and <AssertionURIReference> elements, and elements in other
837 namespaces, with lax schema validation in effect for these other elements.

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

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

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

```
844 <element name="Advice" type="saml:AdviceType"/>  
845 <complexType name="AdviceType">  
846   <choice minOccurs="0" maxOccurs="unbounded">  
847     <element ref="saml:AssertionIDReference"/>  
848     <element ref="saml:AssertionURIReference"/>  
849     <element ref="saml:Assertion"/>  
850     <element ref="saml:EncryptedAssertion"/>  
851     <any namespace="##other" processContents="lax"/>  
852   </choice>  
853 </complexType>
```

854 2.4.4 Element <EncryptedAssertion>

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

858 <xenc:EncryptedData> [Required]

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

863 <xenc:EncryptedKey> [Zero or more]

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

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

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

```
872 <element name="EncryptedAssertion" type="saml:EncryptedAssertionType"/>  
873 <complexType name="EncryptedAssertionType">
```

```

874     <sequence>
875         <element ref="xenc:EncryptedData" />
876         <element ref="xenc:EncryptedKey" minOccurs="0"
877 maxOccurs="unbounded" />
878     </sequence>
879 </complexType>

```

880 2.5 Statements

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

882 2.5.1 Element <Statement>

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

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

```

888 <element name="Statement" type="saml:StatementAbstractType" />
889 <complexType name="StatementAbstractType" abstract="true" />

```

890 2.5.2 Element <AuthnStatement>

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

895 **AuthnInstant** [Required]

896 Specifies the time at which the authentication took place. The time value is encoded in UTC as
897 described in Section Time Values.

898 | **SessionIndex** [RequiredOptional]

899 Indexes a particular session between the subject and the authority issuing this statement. The value
900 of the attribute SHOULD be a small, positive integer, but may be any string of text. This value MUST
901 NOT be a unique value identifying a principal's session at the authority.

902 **SessionNotOnOrAfter** [Optional]

903 Specifies a time instant at which the session between the subject and the authority issuing this
904 statement MUST be considered ended. The time value is encoded in UTC as described in Section
905 Time Values.

906 <SubjectLocality> [Optional]

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

909 | <AuthnContext> [OptionalRequired]

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

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

916 Assertions containing <AuthnStatement> elements MUST contain a <Subject> element.

917 The following schema fragment defines the <AuthnStatement> element and its **AuthnStatementType**
918 complex type:

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

936 2.5.2.1 Element <SubjectLocality>

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

939 ~~IP~~Address [Optional]

940 The ~~IP~~network address of the system from which the subject was authenticated.

941 DNS~~Address~~Name [Optional]

942 The DNS ~~address~~name of the system from which the subject was authenticated.

943 This element is entirely advisory, since both these fields are quite easily “spoofed,” but ~~current practice~~
944 ~~appears to require its inclusion may be useful information in some applications.~~

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

```
947 <element name="SubjectLocality"
948 type="saml:SubjectLocalityType"/>
949 <complexType name="SubjectLocalityType">
950   <attribute name="IPAddress" type="string" use="optional"/>
951   <attribute name="DNSAddressName" type="string" use="optional"/>
952 </complexType>
```

953 2.5.2.2 Element <AuthnContext>

954 The <AuthnContext> element specifies the context of an authentication event with a context class
955 reference, a context statement or statement reference, or both. It's complex **AuthnContextType** has the
956 following elements:

957 <AuthnContextClassRef> [Optional]

958 A URI identifying an authentication context class that describes the authentication context statement
959 that follows.

960 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]
 961 Either an authentication context declaration, or a URI that identifies such a declaration. The URI MAY
 962 directly resolve into an XML document containing the referenced declaration.

963 <AuthenticatingAuthority> [Zero or More]
 964 Zero or more unique identifiers of authentication authorities that were involved in the authentication of
 965 the principal in addition to the assertion issuer.

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

```

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

984 2.5.3 Element <AttributeStatement>

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

988 <Attribute> or <EncryptedAttribute> [One or More]
 989 The <Attribute> element specifies an attribute of the subject. An encrypted SAML attribute may be
 990 included with the <EncryptedAttribute> element.

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

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

```

994 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
995 <complexType name="AttributeStatementType">
996   <complexContent>
997     <extension base="saml:StatementAbstractType">
998       <choice maxOccurs="unbounded">
999         <element ref="saml:Attribute"/>
1000         <element ref="saml:EncryptedAttribute"/>
1001       </choice>
1002     </extension>
1003   </complexContent>
1004 </complexType>
  
```

1005 2.5.3.1 Elements <AttributeDesignator> and <Attribute>

1006 The <AttributeDesignator> element identifies an attribute by name within an attribute namespace. It
 1007 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that the values

1008 | of specific SAML attributes values within a specific namespace be returned (see Section Element
1009 | <AttributeQuery> for more information). The <AttributeDesignator> element contains the following
1010 | XML attributes:

1011 | Name [Required]

1012 | The name of the attribute.

1013 | NameFormat [Optional]

1014 | A URI reference representing the classification of the attribute name for purposes of interpreting
1015 | the name. See Section 8.4 for some URI references that MAY be used as the value of the
1016 | NameFormat attribute and their associated descriptions and processing rules. If no NameFormat
1017 | value is provided, the identifier urn:oasis:names:tc:SAML:2.0:attrname-
1018 | format:unspecified (see Section 8.4.1) is in effect.

1019 | FriendlyName [Optional]

1020 | A string that provides a more human-readable form of the attribute's name, which may be useful
1021 | in cases in which the actual Name is complex, such as an OID or a UUID. This value MUST NOT
1022 | be used as a basis for formally identifying SAML attributes.

1023 | Arbitrary attributes

1024 | This complex type uses an <xsd:anyAttribute> extension point to allow for arbitrary XML
1025 | attributes to be added to <AttributeDesignator> constructs without the need for an explicit
1026 | schema extension. This allows additional fields to be added as needed to supply additional
1027 | parameters to be used in an attribute query. SAML extensions MUST NOT add local (non-
1028 | namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace
1029 | to the AttributeType complex type or to any element bound to this type or a derivation of it; such
1030 | attributes are reserved for future maintenance and enhancement of SAML itself.

1031 | The following schema fragment defines the <AttributeDesignator> element and its
1032 | **AttributeDesignatorType** complex type:

```
1033 | <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
1034 | <complexType name="AttributeDesignatorType">  
1035 |   <attribute name="Name" type="string" use="required"/>  
1036 |   <attribute name="NameFormat" type="anyURI" use="optional"/>  
1037 |   <attribute name="FriendlyName" type="string" use="optional"/>  
1038 |   <anyAttribute namespace="##other" processContents="lax"/>  
1039 | </complexType>
```

1040 | **2.5.3.2 Element <Attribute>**

1041 | The <Attribute> element supplies the value for an attribute of an assertion subject. It has the
1042 | **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following
1043 | element and attributes:

1044 | <AttributeValue> [Any Number]

1045 | The value of the attribute. If an attribute contains more than one discrete value, it is RECOMMENDED
1046 | that each value appear in its own <AttributeValue> element. If the attribute exists but has no
1047 | value, then the <AttributeValue> element MUST be omitted. If more than one
1048 | <AttributeValue> element is supplied for an attribute, and any of the elements have a datatype
1049 | assigned through xsi:type, then all of the <AttributeValue> elements must have the identical
1050 | datatype assigned.

1051 Arbitrary attributes

1052 This complex type inherits from **AttributeDesignatorType** the ability to add arbitrary XML
1053 attributes to <Attribute> constructs without the need for an explicit schema extension. This
1054 allows additional fields to be added as needed to supply the context in which the attribute should
1055 be understood. SAML extensions MUST NOT add local (non-namespace-qualified) XML
1056 attributes [or XML attributes qualified by a SAML-defined namespace](#) to the AttributeType complex
1057 type or to any element bound to this type or a derivation of it; such attributes are reserved for
1058 future maintenance and enhancement of SAML itself.

1059 The following schema fragment defines the <Attribute> element and its **AttributeType** complex type:

```
1060 <element name="Attribute" type="saml:AttributeType"/>
1061 <complexType name="AttributeType">
1062   <complexContent>
1063     <extension base="saml:AttributeDesignatorType">
1064       <sequence>
1065         <element ref="saml:AttributeValue" minOccurs="0"
1066 maxOccurs="unbounded"/>
1067       </sequence>
1068     </extension>
1069   </complexContent>
1070 </complexType>
```

1071 2.5.3.2.1 Element <AttributeValue>

1072 The <AttributeValue> element supplies the value of a specified attribute. It is of the **anyType** type,
1073 which allows any well-formed XML to appear as the content of the element.

1074 If the data content of an AttributeValue element is of an XML Schema simple type (such as **xsd:integer** or
1075 **xsd:string**), the data-type MAY be declared explicitly by means of an `xsi:type` declaration in the
1076 <AttributeValue> element. If the attribute value contains structured data, the necessary data
1077 elements MAY be defined in an extension schema.

1078 **Note:** Specifying a datatype on <AttributeValue> using `xsi:type` will require the
1079 presence of the extension schema that defines the datatype in order for schema
1080 processing to proceed.

1081 The following schema fragment defines the <AttributeValue> element:

```
1082 <element name="AttributeValue" type="anyType"/>
```

1083 2.5.3.3 Element <EncryptedAttribute>

1084 The <EncryptedAttribute> element represents a SAML attribute in encrypted fashion, as defined by
1085 the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAttribute>
1086 element contains the following elements:

1087 <xenc:EncryptedData> [Required]

1088 The encrypted content and associated encryption details, as defined by the XML Encryption
1089 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if
1090 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The
1091 encrypted content MUST contain an element that has a type that is derived from **AttributeType**.

1092 <xenc:EncryptedKey> [Zero or more]

1093 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
1094 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of

1095 the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity as defined by
1096 section 8.3.6.

1097 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through
1098 an intermediary.

1099 The following schema fragment defines the `<EncryptedAttribute>` element and its
1100 **EncryptedAttributeType** complex type:

```
1101 <element name="EncryptedAttribute" type="saml:EncryptedAttributeType" />  
1102 <complexType name="EncryptedAttributeType" >  
1103   <sequence>  
1104     <element ref="xenc:EncryptedData" />  
1105     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded" />  
1106   </complexType>
```

1107 2.5.4 Element `<AuthzDecisionStatement>`

1108 **Note:** The `<AuthzDecisionStatement>` feature has been frozen as of SAML V2.0,
1109 with no future enhancements planned. Users who require additional functionality may
1110 want to consider the eXtensible Access Control Markup Language [XACML], which offers
1111 enhanced authorization decision features.

1112 The `<AuthzDecisionStatement>` element describes a statement by the SAML authority asserting that
1113 a request for access by the statement's subject to the specified resource has resulted in the specified
1114 authorization decision on the basis of some optionally specified evidence.

1115 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1116 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
1117 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
1118 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1119 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1120 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1121 dependent. When a scheme uses elements of the common syntax, it will also use the common
1122 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1123 with an explicit `":port"`, where the port is the default for the scheme, is equivalent to one where
1124 the port is elided.

1125 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the
1126 URI normalized form wherever possible as follows:

- 1127 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1128 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1129 Inconsistent URI reference interpretation can also result from differences between the URI reference
1130 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1131 employed to specify an access control policy language. The following security conditions should be
1132 satisfied by the system which employs SAML assertions:

- 1133 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive, a
1134 requester SHOULD NOT be able to gain access to a denied resource by changing the case of a part
1135 of the resource URI reference.
- 1136 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users to
1137 establish logical equivalences between file system entries. A requester SHOULD NOT be able to gain
1138 access to a denied resource by creating such an equivalence.

1139 The <AuthzDecisionStatement> element is of type **AuthzDecisionStatementType**, which extends
1140 **StatementAbstractType** with the addition of the following elements (in order) and attributes:

1141 Resource [Required]

1142 A URI reference identifying the resource to which access authorization is sought. It is permitted for
1143 this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the
1144 start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1145 Decision [Required]

1146 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1147 the **DecisionType** simple type.

1148 <Action> [One or more]

1149 The set of actions authorized to be performed on the specified resource.

1150 <Evidence> [Optional]

1151 A set of assertions that the SAML authority relied on in making the decision.

1152 Assertions containing <AuthzDecisionStatement> elements MUST contain a <Subject> element.

1153 The following schema fragment defines the <AuthzDecisionStatement> element and its
1154 **AuthzDecisionStatementType** complex type:

```
1155 <element name="AuthzDecisionStatement"  
1156 type="saml:AuthzDecisionStatementType"/>  
1157 <complexType name="AuthzDecisionStatementType">  
1158   <complexContent>  
1159     <extension base="saml:StatementAbstractType">  
1160       <sequence>  
1161         <element ref="saml:Action" maxOccurs="unbounded"/>  
1162         <element ref="saml:Evidence" minOccurs="0"/>  
1163       </sequence>  
1164       <attribute name="Resource" type="anyURI" use="required"/>  
1165       <attribute name="Decision" type="saml:DecisionType"  
1166 use="required"/>  
1167     </extension>  
1168   </complexContent>  
1169 </complexType>
```

1170 2.5.4.1 Element <Action>

1171 The <Action> element specifies an action on the specified resource for which permission is sought. It
1172 has the following attribute and string-data content:

1173 Namespace [Optional]

1174 A URI reference representing the namespace in which the name of the specified action is to be
1175 interpreted. If this element is absent, the namespace
1176 **urn:oasis:names:tc:SAML:1.0:urn:oasis:names:tc:SAML:1.0:action:rwdc-negation**
1177 specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

1178 *string data* [Required]

1179 An action sought to be performed on the specified resource.

1180 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```
1181 <element name="Action" type="saml:ActionType"/>  
1182 <complexType name="ActionType">  
1183   <simpleContent>
```

```

1184         <extension base="string">
1185             <attribute name="Namespace" type="anyURI"/>
1186         </extension>
1187     </simpleContent>
1188 </complexType>

```

1189 2.5.4.2 Element <Evidence>

1190 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on
1191 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one
1192 or more of the following elements:

1193 | <AssertionIDReference> [Any number]

1194 | Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1195 | <AssertionURIReference> [Any number]

1196 | Specifies an assertion by reference to a URI.

1197 | <Assertion> [Any number]

1198 | Specifies an assertion by value.

1199 | <EncryptedAssertion> [Any number]

1200 | Specifies an encrypted assertion by value.

1201 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1202 and the SAML authority making the authorization decision. For example, in the case that the SAML relying
1203 party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1204 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1205 assertion as valid either to the relying party or any other third party.

1206 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```

1207 <element name="Evidence" type="saml:EvidenceType"/>
1208 <complexType name="EvidenceType">
1209     <choice maxOccurs="unbounded">
1210         <element ref="saml:AssertionIDReference"/>
1211         <element ref="saml:AssertionURIReference"/>
1212         <element ref="saml:Assertion"/>
1213         <element ref="saml:EncryptedAssertion"/>
1214     </choice>
1215 </complexType>

```

3 SAML Protocols

1216

1217 SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of
1218 protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting
1219 queries, assertions, and other messages using existing widely deployed transport protocols.

1220 Specific SAML request and response messages derive from common types. The requester sends an
1221 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1222 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1223



1225

Figure 1: SAML Request-Response Protocol

1226 The protocols defined by SAML achieve the following actions:

- 1227 • Returning one or more requested assertions (includes a direct request of the desired assertions, as
1228 well as querying for assertions that meet particular criteria)
- 1229 • Performing authentication on request and returning the corresponding assertion
- 1230 • Registering a name identifier or terminating a name registration on request
- 1231 • Retrieve a protocol message that has been requested by means of an artifact
- 1232 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on
1233 request
- 1234 • Providing a name identifier mapping on request

3.1 Schema Header and Namespace Declarations

1235

1236 The following schema fragment defines the XML namespaces and other header information for the
1237 protocol schema:

1238

```
1239 <schema  
1240     targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1241     xmlns="http://www.w3.org/2001/XMLSchema"  
1242     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1243     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1244     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1245     elementFormDefault="unqualified"  
1246     attributeFormDefault="unqualified"  
1247     blockDefault="substitution"  
1248     version="2.0">  
1249     <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
1250             schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>  
1251     <import namespace="http://www.w3.org/2000/09/xmldsig#"  
1252             schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
1253             schema.xsd"/>  
1254     <annotation>  
1255         <documentation>  
1256             Document identifier: sstc-saml-schema-protocol-2.0
```

```
1256         Location: http://www.oasis-
1257 open.org/committees/documents.php?wg_abbrev=security
1258         </documentation>
1259     </annotation>
1260 ...
1261 </schema>
```

1262 3.2 Requests and Responses

1263 The following sections define the SAML constructs that underlie request and response messages.

1264 3.2.1 Complex Type RequestAbstractType

1265 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1266 This type defines common attributes and elements that are associated with all SAML requests:

1267 **ID** [Required]

1268 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in
1269 Section 1.2.3 for identifier uniqueness. The values of the **ID** attribute in a request and the
1270 **InResponseTo** attribute in the corresponding response MUST match.

1271 **MajorVersion** [Required]

1272 The major version of this request. The identifier for the version of SAML defined in this specification is
1273 2. SAML versioning is discussed in Section SAML Versioning.

1274 **MinorVersion** [Required]

1275 The minor version of this request. The identifier for the version of SAML defined in this specification is
1276 0. SAML versioning is discussed in Section SAML Versioning.

1277 **IssueInstant** [Required]

1278 The time instant of issue of the request. The time value is encoded in UTC as described in Section
1279 Time Values.

1280 **Consent** [Optional]

1281 Indicates whether or not consent has been obtained from a user in the sending this request.

1282 **<Issuer>** [Optional]

1283 Identifies the entity that generated the request message.

1284 **<ds:Signature>** [Optional]

1285 An XML Signature that authenticates the request, as described in Section SAML and XML Signature
1286 Syntax and Processing.

1287 **<Extensions>** [Optional]

1288 This contains optional protocol message extension elements that are agreed upon between the
1289 communicating parties. **SAML extensions MUST NOT include local (non-namespace-qualified)**
1290 **elements or elements qualified by a SAML-defined namespace within this element.**

1291 **Note:** The **<RespondWith>** element has been removed from **<Request>** for V2.0 of
1292 SAML.

1293 **If a **<ds:Signature>** element is present, a responder SHOULD verify that the signature is valid. If it is**
1294 **invalid, the responder SHOULD NOT rely on the contents of the request and SHOULD respond with an**
1295 **error.**

1296 The following schema fragment defines the **RequestAbstractType** complex type:

```
1297 <complexType name="RequestAbstractType" abstract="true">
1298   <sequence>
1299     <element ref="saml:Issuer" minOccurs="0"/>
1300     <element ref="ds:Signature" minOccurs="0"/>
1301
1302     <element ref="samlp:Extensions" minOccurs="0"/>
1303   </sequence>
1304   <attribute name="ID" type="ID" use="required"/>
1305   <attribute name="MajorVersion" type="integer" use="required"/>
1306   <attribute name="MinorVersion" type="integer" use="required"/>
1307   <attribute name="IssueInstant" type="dateTime" use="required"/>
1308   <attribute name="Consent" type="anyURI" use="optional"/>
1309 </complexType>
1310 <element name="Extensions" type="samlp:ExtensionsType"/>
1311 <complexType name="ExtensionsType">
1312   <sequence>
1313     <any namespace="##other" processContents="lax"
1314     maxOccurs="unbounded"/>
1315   </sequence>
1316 </complexType>
```

1317 3.2.1.1 Complex Type StatusResponseType

1318 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
1319 defines common attributes and elements that are associated with all SAML responses:

1320 ID [Required]

1321 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in
1322 Section 1.2.3 for identifier uniqueness.

1323 InResponseTo [Optional]

1324 A reference to the identifier of the request to which the response corresponds, if any. If the response
1325 is not generated in response to a request, or if the ID attribute value of a request cannot be
1326 determined (because the request is malformed), then this attribute MUST NOT be present. Otherwise,
1327 it MUST be present and its value MUST match the value of the corresponding request's ID attribute
1328 value.

1329 MajorVersion [Required]

1330 The major version of this response. The identifier for the version of SAML defined in this specification
1331 is 2. SAML versioning is discussed in Section SAML Versioning.

1332 MinorVersion [Required]

1333 The minor version of this response. The identifier for the version of SAML defined in this specification
1334 is 0. SAML versioning is discussed in Section SAML Versioning.

1335 IssueInstant [Required]

1336 The time instant of issue of the response. The time value is encoded in UTC as described in Section
1337 Time Values.

1338 Recipient [Optional]

1339 The intended recipient of this response. This is useful to prevent malicious forwarding of responses to
1340 unintended recipients, a protection that is required by some use profiles. It is set by the generator of
1341 the response to a URI reference that identifies the intended recipient. If present, the actual recipient
1342 MUST check that the URI reference identifies the recipient or a resource managed by the recipient. If
1343 it does not, the response MUST be discarded.

- 1344 <Issuer> [Optional]
 1345 Identifies the entity that generated the response message.
- 1346 <ds:Signature> [Optional]
 1347 An XML Signature that authenticates the response, as described in Section SAML and XML Signature
 1348 Syntax and Processing.
- 1349 <Extensions> [Optional]
 1350 This contains optional protocol message extension elements that are agreed upon between the
 1351 communicating parties. SAML extensions MUST NOT include local (non-namespace-qualified)
 1352 elements or elements qualified by a SAML-defined namespace within this element.
- 1353 <Status> [Required]
 1354 A code representing the status of the corresponding request.
- 1355 If a <ds:Signature> element is present, a requester SHOULD verify that the signature is valid. If it is
 1356 invalid, the requester SHOULD NOT rely on the contents of the response.

1357 The following schema fragment defines the **StatusResponseType** complex type:

```

1358 <complexType name="StatusResponseType">
1359   <sequence>
1360     <element ref="saml:Issuer" minOccurs="0"/>
1361     <element ref="ds:Signature" minOccurs="0"/>
1362     <element ref="samlp:Extensions" minOccurs="0"/>
1363     <element ref="samlp:Status"/>
1364   </sequence>
1365   <attribute name="ID" type="ID" use="required"/>
1366   <attribute name="InResponseTo" type="NCName" use="optional"/>
1367   <attribute name="MajorVersion" type="integer" use="required"/>
1368   <attribute name="MinorVersion" type="integer" use="required"/>
1369   <attribute name="IssueInstant" type="dateTime" use="required"/>
1370   <attribute name="Recipient" type="anyURI" use="optional"/>
1371 </complexType>

```

1372 3.2.1.2 Element <Status>

1373 The <Status> element contains the following elements:

- 1374 <StatusCode> [Required]
 1375 A code representing the status of the corresponding request.
- 1376 <StatusMessage> [Optional]
 1377 A message which MAY be returned to an operator.
- 1378 <StatusDetail> [Optional]
 1379 Additional information concerning an error condition.

1380 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1381 <element name="Status" type="samlp:StatusType"/>
1382 <complexType name="StatusType">
1383   <sequence>
1384     <element ref="samlp:StatusCode"/>
1385     <element ref="samlp:StatusMessage" minOccurs="0"/>
1386     <element ref="samlp:StatusDetail" minOccurs="0"/>
1387   </sequence>
1388 </complexType>

```

1389 3.2.1.3 Element <StatusCode>

1390 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the
1391 corresponding request. The <StatusCode> element has the following element and attribute:

1392 Value [Required]

1393 The status code value. This attribute contains ~~an XML Schema QName; a namespace prefix MUST~~
1394 ~~be provided~~URI. The value of the topmost <StatusCode> element MUST be from the top-level list
1395 provided in this section.

1397 <StatusCode> [Optional]

1398 A subordinate status code that provides more specific information on an error condition.

1399 The permissible top-level <StatusCode> values are QnamesURI values associated with the SAML
1400 protocol namespace. The local parts of these QNames are defined as follows:

1402 urn:oasis:names:tc:SAML:2.0:status:Success

1403 The request succeeded.

1404 urn:oasis:names:tc:SAML:2.0:status:Requester

1405 The request could not be performed due to an error on the part of the requester.

1406 urn:oasis:names:tc:SAML:2.0:status:Responder

1407 The request could not be performed due to an error on the part of the SAML responder or SAML
1408 authority.

1409 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1410 The SAML responder could not process the request because the version of the request message was
1411 incorrect.

1412 The following second-level status codes are referenced at various places in the specification. Additional
1413 second-level status codes MAY be defined in future versions of the SAML specification.

1414 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1415 The responding provider was unable to successfully authenticate the principal.

1416 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1417 The responding provider does not support the specified name identifier format for the requested
1418 subject.

1419 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1420 The specified authentication context requirements cannot be met by the responder.

1421 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1422 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an
1423 <IDPList> can be resolved or that none of the supported identity providers are available.

1424 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1425 Indicates the identity provider cannot authenticate the principal passively, as has been requested.

1426 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP

1427 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1428 supported by the intermediary.

1427 | [urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded](#)

1428 | Indicates that an identity provider cannot authenticate the principal directly and is not permitted to
1429 | proxy the request further.

1430 | [urn:oasis:names:tc:SAML:2.0:status:RequestDenied](#)

1431 | The SAML responder or SAML authority is able to process the request but has chosen not to respond.
1432 | This status code MAY be used when there is concern about the security context of the request
1433 | message or the sequence of request messages received from a particular requester.

1434 | [urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported](#)

1435 | The SAML responder or SAML authority does not support the request.

1436 | [urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated](#)

1437 | The SAML responder can not process any requests with the protocol version specified in the request.

1438 | [urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh](#)

1439 | The SAML responder cannot process the request because the protocol version specified in the
1440 | request message is a major upgrade from the highest protocol version supported by the responder.

1441 | [urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow](#)

1442 | The SAML responder cannot process the request because the protocol version specified in the
1443 | request message is too low.

1444 | [urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized](#)

1445 | The SAML authority does not wish to support resource-specific attribute queries, or the resource value
1446 | provided in the request message is invalid or unrecognized.

1447 | [urn:oasis:names:tc:SAML:2.0:status:TooManyResponses](#)

1448 | The response message would contain more elements than the SAML responder will return.

1449 | [urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal](#)

1450 | The responding provider does not recognize the principal specified or implied by the request.

1451 | [urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding](#)

1452 | The SAML responder cannot properly fulfill the request using the protocol binding specified in the
1453 | request.

1454 | SAML system entities are free to define more specific status codes ~~in other namespaces, but MUST NOT~~
1455 | ~~define additional codes in the SAML assertion or protocol namespace by defining appropriate URI values.~~
1456 | ~~The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value~~
1457 | ~~attribute and have the above semantics only in that context.~~

1458 | The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
1459 | type:

```

1460 | <element name="StatusCode" type="samlp:StatusCodeType"/>
1461 | <complexType name="StatusCodeType">
1462 |   <sequence>
1463 |     <element ref="samlp:StatusCode" minOccurs="0"/>
1464 |   </sequence>
1465 |   <attribute name="Value" type="@NameanyURI" use="required"/>
1466 | </complexType>

```

1467 **3.2.1.4 Element <StatusMessage>**

1468 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1469 The following schema fragment defines the <StatusMessage> element:

```
1470 <element name="StatusMessage" type="string"/>
```

1471 **3.2.1.5 Element <StatusDetail>**

1472 The <StatusDetail> element MAY be used to specify additional information concerning an error
1473 condition.

1474 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1475 complex type:

```
1476 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1477 <complexType name="StatusDetailType">  
1478   <sequence>  
1479     <any namespace="##any" processContents="lax" minOccurs="0"  
1480     maxOccurs="unbounded"/>  
1481   </sequence>  
1482 </complexType>
```

1483 **3.3 Assertion Query and Request Protocol**

1484 This section defines messages and processing rules for requesting existing assertions by reference or
1485 querying for assertions by subject and statement type.

1486 **3.3.1 Element <AssertionIDRequest>**

1487 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>
1488 message can be used to request that the assertion(s) be returned in a <Response> message. The
1489 <saml:AssertionIDReference> element is used to specify the assertion(s) to return. See Section
1490 Element <AssertionIDReference> for more information on this element.

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

```
1492 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>  
1493 <complexType name="AssertionIDRequestType">  
1494   <complexContent>  
1495     <extension base="samlp:RequestAbstractType">  
1496       <sequence>  
1497         <element ref="saml:AssertionIDReference"  
1498         maxOccurs="unbounded"/>  
1499       </sequence>  
1500     </extension>  
1501   </complexContent>  
1502 </complexType>
```

1503 **3.3.2 Queries**

1504 The following sections define the SAML query request messages.

1505 3.3.2.1 Element <SubjectQuery>

1506 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1507 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and
1508 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the <Subject>
1509 element to **RequestAbstractType**.

1510 The following schema fragment defines the <SubjectQuery> element and its
1511 **SubjectQueryAbstractType** complex type:

```
1512 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1513 <complexType name="SubjectQueryAbstractType" abstract="true">  
1514   <complexContent>  
1515     <extension base="samlp:RequestAbstractType">  
1516       <sequence>  
1517         <element ref="saml:Subject"/>  
1518       </sequence>  
1519     </extension>  
1520   </complexContent>  
1521 </complexType>
```

1522 3.3.2.2 Element <AuthnQuery>

1523 The <AuthnQuery> message element is used to make the query "What assertions containing
1524 authentication statements are available for this subject?" A successful <Response> will contain one or
1525 more assertions containing authentication statements.

1526 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using
1527 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts
1528 that have occurred in a previous interaction between the indicated subject and the Authentication
1529 Authority.

1530 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1531 the following element and attribute:

1532 <RequestedAuthnContext> [Optional]

1533 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1534 containing authentication statements do you have for this subject that satisfy the authentication
1535 context requirements in this element?"

1536 SessionIndex [Optional]

1537 If present, specifies a filter for possible responses. Such a query asks the question "What assertions
1538 containing authentication statements do you have for this subject within the context of the supplied
1539 session information?"

1540 In response to an authentication query, a SAML authority returns assertions with authentication
1541 statements as follows:

- 1542 • Rules given in [Section 3.3.4](#) for matching against the <Subject> element of the query identify the
1543 assertions that may be returned.
- 1544 • If the SessionIndex attribute is present in the query, at least one <AuthnStatement> element in
1545 the set of returned assertions MUST contain an SessionIndex attribute that matches the
1546 SessionIndex attribute in the query. It is OPTIONAL for the complete set of all such matching
1547 assertions to be returned in the response.
- 1548 • If the <RequestedAuthnContext> element is present in the query, at least one
1549 <AuthnStatement> element in the set of returned assertions MUST contain an <AuthnContext>

1550 element that satisfies the element in the query (see section 3.3.2.3). It is OPTIONAL for the complete
1551 set of all such matching assertions to be returned in the response.

1552

1553

1554 The following schema fragment defines the <AuthenticationQuery> element and its
1555 **AuthnQueryType** complex type:

```
1556 <element name="AuthnQuery" type="samlp:AuthnQueryType"/>  
1557 <complexType name="AuthnQueryType">  
1558   <complexContent>  
1559     <extension base="samlp:SubjectQueryAbstractType">  
1560       <attribute name="SessionIndex" type="string"  
1561 use="optional"/>  
1562     </extension>  
1563   </complexContent>  
1564 </complexType>
```

1565 3.3.2.3 Element <RequestedAuthnContext>

1566 The <RequestedAuthnContext> element specifies the authentication context requirements of
1567 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**
1568 complex type defines the following elements and attributes:

1569 <AuthnContextClassRef> or <AuthnContextDeclRef> [One or More]

1570 Specifies one or more URIs identifying authentication context classes or declarations.

1571 Comparison [Optional]

1572 Specifies the comparison method used to evaluate the requested context classes or statements, one
1573 of "exact", "minimum", "maximum", or "better". The default is "exact".

1574 Either a set of class references or declaration references can be used. Additionally, the set of supplied
1575 references MUST be evaluated as an ordered set, where the first element is the most preferred
1576 authentication context class or declaration. If none of the specified classes or declarations can be satisfied
1577 in accordance with the rules below, then the responder MUST return a <Response> message with a
1578 second-level <StatusCode> of
1579 [samlp:urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext](#).

1580 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
1581 statement MUST be the exact match of at least one of the authentication contexts specified.

1582 If Comparison is set to "minimum", then the resulting authentication context in the authentication
1583 statement MUST be at least as strong (as deemed by the responder) as one of the authentication
1584 contexts specified.

1585 If Comparison is set to "better", then the resulting authentication context in the authentication statement
1586 MUST be stronger (as deemed by the responder) than any one of the authentication contexts specified.

1587 If Comparison is set to "maximum", then the resulting authentication context in the authentication
1588 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength
1589 of at least one of the authentication contexts specified.

1590 The following schema fragment defines the <RequestedAuthnContext> element and its
1591 **RequestedAuthnContextType** complex type:

```
1592 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>  
1593 <complexType name="RequestedAuthnContextType">
```

```

1594     <choice>
1595         <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1596         <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1597     </choice>
1598     <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1599 use="optional"/>
1600 </complexType>
1601 <simpleType name="AuthnContextComparisonType">
1602     <restriction base="string">
1603         <enumeration value="exact"/>
1604         <enumeration value="minimum"/>
1605         <enumeration value="maximum"/>
1606         <enumeration value="better"/>
1607     </restriction>
1608 </simpleType>

```

1609 3.3.2.4 Element <AttributeQuery>

1610 The <AttributeQuery> element is used to make the query “Return the requested attributes for this
1611 subject.” A successful response will be in the form of assertions containing attribute statements. This
1612 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1613 the following element and attribute:

1614 <AttributeDesignator> [Any Number]

1615 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no
1616 attributes are specified, it indicates that all attributes allowed by policy are requested.

1617 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1618 • Rules given in [Section 3.3.4](#) for matching against the <Subject> element of the query identify the
1619 assertions that may be returned.
- 1620 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute
1621 values returned, as noted above.
- 1622 • The attribute values returned MAY be constrained by application-specific policy considerations.

1623 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1624 complex type:

```

1625 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1626 <complexType name="AttributeQueryType">
1627     <complexContent>
1628         <extension base="samlp:SubjectQueryAbstractType">
1629             <sequence>
1630                 <element ref="saml:AttributeDesignator"
1631 minOccurs="0" maxOccurs="unbounded"/>
1632             </sequence>
1633         </extension>
1634     </complexContent>
1635 </complexType>

```

1636 3.3.2.5 Element <AuthzDecisionQuery>

1637 The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource
1638 be allowed for this subject, given this evidence?” A successful response will be in the form of assertions
1639 containing authorization decision statements.

1640 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no
1641 future enhancements planned. Users who require additional functionality may want to
1642 consider the eXtensible Access Control Markup Language [XACML], which offers
1643 enhanced authorization decision features.

1644 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1645 addition of the following elements and attribute:

1646 Resource [Required]

1647 A URI reference indicating the resource for which authorization is requested.

1648 <Action> [One or More]

1649 The actions for which authorization is requested.

1650 <Evidence> [Optional]

1651 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1652 In response to an authorization decision query, a SAML authority returns assertions with authorization
1653 decision statements as follows:

- 1654 • Rules given in [Section 3.3.4.14](#) for matching against the <Subject> element of the query identify
1655 the assertions that may be returned.

1656 The following schema fragment defines the <AuthzDecisionQuery> element and its
1657 **AuthzDecisionQueryType** complex type:

```
1658 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>  
1659 <complexType name="AuthzDecisionQueryType">  
1660   <complexContent>  
1661     <extension base="samlp:SubjectQueryAbstractType">  
1662       <sequence>  
1663         <element ref="saml:Action" maxOccurs="unbounded"/>  
1664         <element ref="saml:Evidence" minOccurs="0"/>  
1665       </sequence>  
1666       <attribute name="Resource" type="anyURI" use="required"/>  
1667     </extension>  
1668   </complexContent>  
1669 </complexType>
```

1670 3.3.3 Element <Response>

1671 The <Response> message element is used when a response consists of a list of zero or more
1672 assertions that answer the request. It has the complex type **ResponseType**, which extends
1673 **StatusResponseType** by adding the following elements:

1674 <Assertion> or <EncryptedAssertion> [Any Number]

1675 Specifies an assertion by value, or optionally an encrypted assertion by value. (See SectionElement
1676 <Assertion> for more information.)

1677 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1678 <element name="Response" type="samlp:ResponseType"/>  
1679 <complexType name="ResponseType">  
1680   <complexContent>  
1681     <extension base="samlp:StatusResponseType">  
1682       <choice minOccurs="0" maxOccurs="unbounded">  
1683         <element ref="saml:Assertion"/>  
1684         <element ref="saml:EncryptedAssertion"/>  
1685       </choice>
```


1686
1687
1688

```
</extension>  
</complexContent>  
</complexType>
```

1689 3.3.4 Processing Rules

1690 In response to a query message, every assertion returned by a SAML authority MUST contain a
1691 <Subject> element that **strongly matches** the <Subject> element found in the query.

1692 A <Subject> element S1 strongly matches S2 if and only if the following two conditions both apply:

1693 • If S2 includes an identifier element (any element whose type is derived from **BaseIDAbstractType**),
1694 then S1 **mustMUST** include an identical identifier element, but the element MAY be encrypted (or not)
1695 in either S1 or S2. In other words, the decrypted form of the identifier MUST be identical in S1 and S2.
1696 "Identical" means that the identifier element's content and attribute values MUST be the same. An
1697 encrypted identifier will be identical to the original according to this definition, once decrypted.

1698 • If S2 includes one or more <SubjectConfirmation> elements, then S1 **mustMUST** include at
1699 least one <SubjectConfirmation> element such that the assertion's subject can be confirmed in
1700 the manner described by at least one element in the requested set.

1701 As an example of what is and is not permitted, S1 could contain a <NameID> with a particular Format
1702 value, and S2 could contain an <EncryptedID> element that is the result of encrypting S1's <NameID>
1703 element. However, S1 and S2 cannot contain a <NameID> element with different Format values and
1704 element content, even if the two identifiers are considered to refer to the same principal.

1705 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
1706 expressed by a query or assertion reference, the <Response> element MUST NOT contain an
1707 <Assertion> element and MUST include a <StatusCode> element with the value
1708 urn:oasis:names:tc:SAML:2.0:status:Success. It MAY return a <StatusMessage> element
1709 with additional information.

1710 All other processing rules associated with the underlying request and response messages MUST be
1711 observed.

1712 3.4 Authentication Request Protocol

1713 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
1714 authentication statements to establish a security context at one or more relying parties, it can use the
1715 authentication request protocol to send an <AuthnRequest> message to a SAML authority and request
1716 that it return a <Response> message containing one or more such assertions. Such assertions MAY
1717 contain additional statements of any type, but at least one assertion MUST contain at least one
1718 authentication statement. -A SAML authority that supports this protocol is also termed an identity provider.

1719 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
1720 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
1721 are not specified, though the means of authentication MAY impact the content of the response. Other
1722 issues related to the validation of authentication credentials by the identity provider or any communication
1723 between the identity provider and any other entities involved in the authentication process are also out of
1724 scope of this protocol.

1725 The descriptions and processing rules in the following sections reference the following actors, many of
1726 whom might be the same entity in a particular profile of use:

1727 Request Issuer

1728 The entity who creates the authentication request and to whom the response is to be returned.

- 1729 **Presenter**
- 1730 The entity who presents the request to the authority and either authenticates itself during the
1731 sending of the message, or relies on an existing security context to establish its identity. If not the
1732 request issuer, the sender acts as an intermediary between the request issuer and the responding
1733 identity provider.
- 1734 **Requested Subject**
- 1735 The entity about whom one or more assertions are being requested.
- 1736 **Confirming Subject**
- 1737 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`
1738 elements of the resulting assertion(s).
- 1739 **Relying Party**
- 1740 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
1741 the profile or context of use, generally to establish a security context.

1742 **3.4.1 Element `<AuthnRequest>`**

1743 To request that an identity provider issue an authentication assertion, an entity authenticates to it (or relies
1744 on an existing security context) and sends it an `<AuthnRequest>` message that describes the properties
1745 that the resulting assertion needs to have to satisfy its purpose. Among these properties may be
1746 information that relates to the content of the assertion and/or information that relates to how the resulting
1747 `<Response>` message should be delivered to the request issuer.

1748 The request issuer might not be the same as the presenter of the request, if for example the request
1749 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the
1750 requested subject to provide a service.

1751 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
1752 by the protocol binding used to deliver the message.

1753 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
1754 adds the following elements and attributes, all of which are optional in general, but may be required by
1755 specific profiles:

1756 `<Subject>` [Optional]

1757 Specifies the requested subject of the resulting assertion(s). This may include one or more
1758 `<SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions' can
1759 be confirmed.

1760 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
1761 requested subject. If no `<SubjectConfirmation>` elements are included, then the presenter is
1762 presumed to be the only confirming entity required and the method is implied by the profile of use
1763 and/or the policies of the identity provider.

1764 `<NameIDPolicy>` [Optional]

1765 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
1766 then any type of identifier supported by the identity provider for the requested subject can be used,
1767 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

1768 `<Conditions>` [Optional]

1769 Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the
1770 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary.

1771 <RequestedAuthnContext> [Optional]
1772 Specifies the requirements, if any, that the request issuer places on the authentication context that
1773 applies to the responding provider's authentication of the presenter. See section 3.3.2.3 for processing
1774 rules regarding this element.

1775 <Scoping> [Optional]
1776 Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as
1777 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
1778 providers by the responder.

1779 IsPassive [Optional]
1780 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of the
1781 user interface from the request issuer and interact with the presenter in a noticeable fashion. If a value
1782 is not provided, the default is "true".

1783 ForceAuthn [Optional]
1784 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
1785 rely on a previous security context. If a value is not provided, the default is "false". However, if both
1786 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
1787 presenter unless the constraints of IsPassive can be met.

1788 ProtocolBinding [Optional]
1789 A URI that identifies a SAML protocol binding to be used when returning the <Response> message.

1790 AssertionConsumerServiceIndex [Optional]
1791 Indirectly identifies the location to which the <Response> message should be returned to the request
1792 issuer. It applies only to profiles in which the request issuer is different than the presenter. The identity
1793 provider MUST have a trusted means to map the index value in the attribute to a location associated
1794 with the request issuer. [SAMLMetadata] provides one possible mechanism. If omitted, then the
1795 identity provider MUST return the <Response> message to the default location associated with the
1796 request issuer for the profile of use. References one of a set of <AssertionConsumerService>
1797 elements in the request issuer's metadata as the one to which the <Response> should be returned. It
1798 applies only to profiles that specify use of this metadata element, in which the request issuer is
1799 different than the presenter. If omitted, the metadata element labeled with the isDefault attribute
1800 MUST be used with such profiles.

1801 AssertionConsumerServiceURL [Optional]
1802 Specifies by value the location to which the <Response> message MUST be returned to the request
1803 issuer. The responder MUST insure by some means (~~such as metadata~~) that the value specified is in
1804 fact associated with the request issuer. [SAMLMetadata] provides one possible mechanism.

1805 AttributeConsumingServiceIndex [Optional]
1806 Indirectly identifies information associated with the request issuer describing the SAML attributes the
1807 request issuer desires or requires be supplied by the identity provider in the <Response> message.
1808 The identity provider MUST have a trusted means to map the index value in the attribute to
1809 information associated with the request issuer. [SAMLMetadata] provides one possible mechanism.
1810 The identity provider MAY use this information to populate one or more <AttributeStatement>
1811 elements in the assertion(s) it returns.

1812 ProviderName [Optional]
1813 Specifies the human-readable name of the request issuer for use by the presenter's user agent or the
1814 identity provider.

1815 See Section 3.4.1.5 for general processing rules regarding this message.

1816 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**
1817 complex type:

```
1818 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
1819 <complexType name="AuthnRequestType">
1820   <complexContent>
1821     <extension base="samlp:RequestAbstractType">
1822       <sequence>
1823         <element ref="saml:Subject" minOccurs="0"/>
1824         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
1825         <element
1826           ref="saml:Conditions" minOccurs="0"/>
1827         <element ref="samlp:RequestedAuthnContext"
1828           minOccurs="0"/>
1829         <element ref="samlp:Scoping" minOccurs="0"/>
1830       </sequence>
1831       <attribute name="IsPassive" type="boolean"
1832         use="optional"/>
1833       <attribute name="ForceAuthn" type="boolean"
1834         use="optional"/>
1835       <attribute name="ProtocolBinding" type="anyURI"
1836         use="optional"/>
1837       <attribute name="AssertionConsumerServiceIndex"
1838         type="stringunsignedShort" use="optional"/>
1839       <attribute name="AssertionConsumerServiceURL"
1840         type="anyURI" use="optional"/>
1841       <attribute name="AttributeConsumingServiceIndex"
1842         type="unsignedShort" use="optional"/>
1843       <attribute name="ProviderName" type="string"
1844         use="optional"/>
1845     </extension>
1846   </complexContent>
1847 </complexType>
```

1848 3.4.1.1 Element <NameIDPolicy>

1849 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
1850 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

1851 Format [Required]

1852 Specifies the URI of a name identifier format defined in this or another specification (see Section 8.3
1853 for examples).

1854 SPNameQualifier [Optional]

1855 Used with a Format of `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` or
1856 `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`, it optionally specifies that a
1857 federated identifier be returned (or created) in the namespace of a service provider other than the
1858 request issuing service provider, or in the namespace of an affiliation group of service providers.

1859 AllowCreate [Optional]

1860 Used to indicate whether the identity provider is allowed, in the course of fulfilling the request, to
1861 create a new identifier to represent the principal. Defaults to "true". When "false", the request issuer
1862 constrains the identity provider to only issue an assertion to it if an acceptable identifier for the
1863 principal has already been established between them.

1864 When this element is used, if the content is not understood by or acceptable to the identity provider, then
1865 a <Response> MUST be returned with an error <Status>, and MAY containing a second-level
1866 <StatusCode> of samlp:urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy.

1867 A Format of `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates that the
1868 resulting assertion(s) MUST contain `<EncryptedID>` elements instead of plaintext. The underlying name
1869 identifier's unencrypted form can be of any type supported by the identity provider for the requested
1870 subject.

1871 Any Format value (or the omission of this element) MAY result in an `<EncryptedID>` in the resulting
1872 assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

1873 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**
1874 complex type:

```
1875 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>  
1876 <complexType name="NameIDPolicyType">  
1877   <sequence/>  
1878   <attribute name="Format" type="anyURI" use="required"/>  
1879   <attribute name="SPNameQualifier" type="string" use="optional"/>  
1880   <attribute name="AllowCreate" type="boolean" use="optional"/>  
1881 </complexType>
```

1882 | 3.4.1.2

1883 | 3.4.1.3 Element `<Scoping>`

1884 The `<Scoping>` element specifies the identity providers trusted by the request issuer to authenticate the
1885 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
1886 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
1887 elements and attribute:

1888 `<IDPList>` [Optional]

1889 An advisory list of identity providers and associated information that the request issuer deems
1890 acceptable to respond to the request.

1891 `<RequesterID>` [Zero or More]

1892 Identifies the set requesting entities on whose behalf the request issuer is acting. Used to
1893 communicate the chain of request issuers when proxying occurs, as described in section 3.4.1.9. See
1894 section 8.3.6 for a description of such identifiers.

1895 `ProxyCount` [Optional]

1896 Specifies the number of proxying indirections permissible between the identity provider that receives
1897 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of
1898 zero permits no proxying, while omitting this attribute expresses no such restriction.

1899 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
1900 `<Response>` message with an error `<Status>` and a second-level `<StatusCode>` of
1901 `samlpurn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP` or
1902 `samlpurn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP` if it cannot contact or does not
1903 support any of the specified identity providers.

1904 The following schema fragment defines the `<Scoping>` element and its **ScopingType** complex type:

```
1905 <element name="Scoping" type="samlp:ScopingType"/>  
1906 <complexType name="ScopingType">  
1907   <sequence>  
1908     <element ref="samlp:IDPList" minOccurs="0"/>  
1909     <element ref="samlp:RequesterID" minOccurs="0"  
1910     maxOccurs="unbounded"/>  
1911   </sequence>  
1912   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
```

```
1913 </complexType>
1914 <element name="RequesterID" type="anyURI"/>
```

1915 **3.4.1.4 Element <IDPList>**

1916 The <IDPList> element specifies the identity providers trusted by the request issuer to authenticate the
1917 presenter. Its **IDPListType** complex type defines the following elements:

1918 <IDPEntry> [One or More]

1919 Information about a single identity provider

1920 <GetComplete> [Optional]

1921 If the <IDPList> is not complete, this element may specify a URI that resolves to the complete list.

1922 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
1923 <element name="IDPList" type="samlp:IDPListType"/>
1924 <complexType name="IDPListType">
1925   <sequence>
1926     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
1927     <element ref="samlp:GetComplete" minOccurs="0"/>
1928   </sequence>
1929 </complexType>
1930 <element name="GetComplete" type="anyURI"/>
```

1931 **3.4.1.5 Element <IDPEntry>**

1932 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to authenticate
1933 the presenter. Its **IDPEntryType** complex type defines the following elements:

1934 <ProviderID> [Required]

1935 The unique identifier of the identity provider. [See section 8.3.6 for a description of such identifiers.](#)

1936 <Name> [Optional]

1937 A human readable name for the identity provider.

1938 <Loc> [Optional]

1939 The location of a profile-specific endpoint supporting the authentication request protocol. The binding
1940 to be used must be understood from the profile of use.

1941 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
1942 <element name="IDPEntry" type="samlp:IDPEntryType"/>
1943 <complexType name="IDPEntryType">
1944   <sequence/>
1945   <attribute name="ProviderID" type="anyURI" use="required"/>
1946   <attribute name="Name" type="string" use="optional"/>
1947   <attribute name="Loc" type="anyURI" use="optional"/>
1948 </complexType>
```

1949 **3.4.1.6 Processing Rules**

1950 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
1951 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
1952 kinds of input and output are required or prohibited. The following processing rules apply as invariant
1953 behavior across any profile of this protocol exchange. [All other processing rules associated with the](#)
1954 [underlying request and response messages MUST also be observed.](#)

1955 | ~~The recipient MUST validate any signature present on the request or response message.~~

1956 | The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing
1957 | one or more assertions that meet the specifications defined by the request, or with a <Response>
1958 | message containing a <Status> describing the error that occurred. The responder MAY conduct
1959 | additional message exchanges with the ~~request sender presenter~~ as needed to initiate or complete the
1960 | authentication process, subject to the nature of the protocol binding and the authentication mechanism. As
1961 | described in the next section, this includes proxying the request by directing the presenter to another
1962 | identity provider by issuing its own <AuthnRequest> message, so that the resulting assertion can be
1963 | used to authenticate the presenter to the original responder, in effect using SAML as the authentication
1964 | mechanism.

1965 | If the responder is unable to authenticate the presenter or does not recognize the requested subject, it
1966 | MUST return a <Response> with an error <Status>, and MAY return a second-level <StatusCode> of
1967 | ~~samlpurn:oasis:names:tc:SAML:2.0:status:AuthnFailed~~ or
1968 | urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

1969 | If the <Subject> element in the request is present, then the resulting assertions' <Subject> MUST
1970 | **strongly match** the request <Subject>, as described in section 3.3.4.4, except that the identifier MAY
1971 | be in a different format if specified by <NameIDPolicy>. In such a case, the identifier's physical content
1972 | MAY be different, but it MUST refer to the same principal.

1973 | All of the content defined specifically within <AuthnRequest> is optional, although some may be required
1974 | by certain profiles. In the absence of any specific content at all, the following behavior is assumed:

- 1975 | • The assertion(s) returned MUST contain a <Subject> element that represents the presenter. The
1976 | identifier type and format are determined by the identity provider. At least one statement MUST be
1977 | an <AuthnStatement> that describes the authentication performed by the responder or
1978 | authentication service associated with it.
- 1979 | • The request presenter should, to the extent possible, be the only entity able to satisfy the
1980 | <SubjectConfirmation> of the assertion(s). In the case of weaker confirmation methods,
1981 | binding-specific or other mechanisms will be used to help satisfy this requirement.
- 1982 | • The resulting assertion(s) MUST contain an <AudienceRestriction> element referencing the
1983 | request issuer as an acceptable relying party. Other audiences MAY be included as deemed
1984 | appropriate by the identity provider.

1985 | 3.4.1.7 Proxying

1986 | If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or
1987 | cannot directly authenticate him/her, but believes that the presenter has already authenticated to another
1988 | identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
1989 | <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in
1990 | whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
1991 | identity provider.

1992 | Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the
1993 | enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
1994 | proxying provider can issue an assertion of its own in response to the original <AuthnRequest>,
1995 | completing the overall message exchange. Both the proxying and authenticating identity providers MAY
1996 | include constraints on proxying activity in the messages and assertions they issue, as described in
1997 | previous sections, and below.

1998 | The request issuer can influence proxy behavior by including a <Scoping> element where the provider
1999 | sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be
2000 | proxied by including an ordered <IDPList> of preferred providers.

2001 An identity provider can control secondary use of its assertions by proxying identity providers using a
2002 <ProxyRestriction> element in the assertions it issues.

2003 Proxying Processing Rules

2004 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is
2005 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2006 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

2007 An identity provider MUST NOT proxy a request where <ProxyCount> is set to zero. The identity
2008 provider MUST return an error <Status> containing a second-level <samlp:StatusCode> value of
2009 ~~samlpurn:oasis:names:tc:SAML:2.0:status~~:ProxyCountExceeded, unless it can directly
2010 authenticate the presenter.

2011 If it chooses to proxy to a SAML identity provider, when creating the new <AuthnRequest>, anthe
2012 proxying identity provider MUST include equivalent or stricter forms of all the information included in the
2013 original request (such as authentication context policy). Note however that the proxying provider is free to
2014 specify whatever <NameIDPolicy> it wishes to maximize the chances of a succesful response.

2015 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
2016 have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for
2017 example) will be honored by the authenticating provider.

2018 The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less
2019 than the original value. If the original request does not contain a <ProxyCount> attribute, then the new
2020 request SHOULD contain a <ProxyCount> attribute.

2021 If an <IDPList> was specified in the original request, the new request MUST also contain an
2022 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the
2023 <IDPList>, but MUST NOT remove any from the list.

2024 The authentication request and response are processed in normal fashion, in accordance with the rules
2025 given in Section 3.4.1.8 and the profile of use. Once the presenter has authenticated to the proxying
2026 identity provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2027 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
2028 relevant information from the original assertion or non-SAML equivalent. The original assertion will
2029 be restricted by <AudienceRestriction> to (at least) the proxying identity provider, while the
2030 new assertion's condition will reference (at least) the original request issuer.
- 2031 • The new assertion's <Subject> should contain an identifier that satisfies the original request
2032 issuer's preferences, as defined by its <NameIDPolicy> element.
- 2033 • The <AuthnStatement> in the new assertion MUST include an <AuthnContext> element
2034 containing an <ae:AuthenticatingAuthority> element referencing the identity provider to
2035 which the proxying identity provider referred the presenter. If the original assertion contains
2036 <AuthnContext> information that includes one or more <ae:AuthenticatingAuthority>
2037 elements, those elements SHOULD be included in the new assertion, with the new element placed
2038 after them.
- 2039 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
2040 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
2041 consistent over time across different requests. The value MUST not conflict with values used or
2042 generated by other SAML providers.
- 2043 • Any other <AuthnContext> information MAY be copied, translated, or omitted in accordance with
2044 the policies of the proxying identity provider, provided that the original requirements dictated by the
2045 request issuer are met.

2046 If, in the future, the identity provider is asked to authenticate the same presenter for a second request
2047 issuer, and this request is equally or less strict than the original request, the identity provider MAY skip the
2048 creation of a new <AuthnRequest> to the authenticating identity provider and immediately issue another
2049 assertion (assuming the original assertion or non-SAML equivalent it received is still valid). The concrete
2050 definition of "equally or less strict" is up to the proxying identity provider.

2051 **3.5 Artifact Resolution Protocol**

2052 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
2053 transported in a SAML binding by reference instead of by value. Both requests and responses can be
2054 obtained by reference using this specialized protocol. A message sender, instead of binding a message to
2055 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2056 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver
2057 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
2058 protocol to dereference the artifact into the original protocol message. The most common use for
2059 this mechanism is with bindings that cannot easily carry a message because of size constraints.

2060 Depending on the characteristics of the underlying message being passed by reference, the artifact
2061 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2062 confidentiality, etc. from the protocol binding used to dereference the artifact. In all cases, the
2063 artifact MUST exhibit a single-use semantic such that once it has been successfully
2064 dereferenced, it can no longer be used by any party.

2065 Regardless of the protocol message obtained, the result of dereferencing an artifact MUST be
2066 treated exactly as if the message so obtained had been sent originally in place of the artifact.

2067 **3.5.1 Element <ArtifactRequestResolve>**

2068 The <ArtifactRequestResolve> message is used to request that a SAML protocol message be
2069 returned in an <ArtifactResponse> message by specifying an artifact that represents the SAML
2070 protocol message. The original transmission of the artifact is governed by the specific protocol binding or
2071 profile of SAML that is being used; see the SAML specifications for bindings [SAMLBind] and profiles
2072 [SAMLProf] for more information on the use of artifacts in bindings and profiles.

2073 The <ArtifactRequestResolve> message SHOULD be signed or otherwise authenticated and integrity
2074 protected by the protocol binding used to deliver the message.

2075 ~~The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a~~
2076 ~~Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.~~

2077 This message has the complex type **ArtifactRequestResolveType**, which extends **RequestAbstractType**
2078 and adds the following element:

2079 <Artifact> [Required]

2080 The artifact value that the requester received and now wishes to translate into the protocol message it
2081 represents. See [SAMLBind] for specific artifact format information.

2082 The following schema fragment defines the <ArtifactRequestResolve> element and its
2083 **ArtifactRequestResolveType** complex type:

```
2084 <element name="ArtifactRequestResolve" type="saml:ArtifactRequestResolveType"/>  
2085 <complexType name="ArtifactRequestResolveType">  
2086   <complexContent>  
2087     <extension base="saml:RequestAbstractType">  
2088       <sequence>  
2089         <element ref="saml:Artifact"/>  
2090       </sequence>
```

```

2091         </extension>
2092     </complexContent>
2093 </complexType>
2094 <element name="Artifact" type="string"/>

```

2095 3.5.2 Element <ArtifactResponse>

2096 | The recipient of an <ArtifactRequestsolve> message MUST respond with an
2097 | <ArtifactResponse> message, which is of complex type **ArtifactResponseType**, which extends
2098 | **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2099 | message being returned. This wrapped message element can be a request or a response.

2100 | The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2101 | protected by the protocol binding used to deliver the message.

2102 | ~~The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a
2103 | Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.~~

2104 | The following schema fragment defines the <ArtifactResponse> element and its
2105 | **ArtifactResponseType** complex type:

```

2106 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2107 <complexType name="ArtifactResponseType">
2108     <complexContent>
2109         <extension base="samlp:StatusResponseType">
2110             <sequence>
2111                 <any namespace="##any" processContents="lax"
2112 minOccurs="0"/>
2113             </sequence>
2114         </extension>
2115     </complexContent>
2116 </complexType>

```

2117 3.5.3 Processing Rules

2118 | ~~The recipient MUST validate any signature present on the request or response message.~~

2119 | If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2120 | an <ArtifactResponse> message. Otherwise, it responds with an <ArtifactResponse> message
2121 | with no embedded message. In both cases, the <Status> element MUST include a <StatusCode>
2122 | element with the code value *Success*. A response message with no embedded message inside it is
2123 | termed an empty response in the remainder of this section.

2124 | The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent
2125 | request with the same artifact by any requester results in an empty response as described above.

2126 | Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY
2127 | be intended for consumption by any party that receives it and can respond appropriately. In most other
2128 | cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2129 | be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2130 | receives an <ArtifactRequestsolve> message from a requester that cannot authenticate itself as the
2131 | original intended recipient, then the artifact issuer MUST return an empty response.

2132 | The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2133 | that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2134 | return it in an <ArtifactRequestsolve> message to the issuer.

2135 | Note that the <ArtifactResponse>' message's InResponseTo attribute MUST contain the value of
2136 | the corresponding <ArtifactRequestsolve> message's ID attribute, but the embedded protocol

2137 message will contain its own message identifier, and in the case of an embedded response, may contain
2138 a different `InResponseTo` value that corresponds to the original request message to which the
2139 embedded message is responding.

2140 All other processing rules associated with the underlying request and response messages MUST be
2141 observed.

2142 3.6 Name Identifier Management Protocol

2143 After establishing a persistent name identifier for a principal, an identity provider wishing to change the
2144 value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name
2145 identifier will no longer be used to refer to the principal, informs service providers of the change by
2146 sending them a `<ManageNameIDRequest>` message.

2147 The same message MAY also be used by a service provider to register or change the `SPProvidedID`
2148 value to be included when the underlying name identifier is used to communicate with it, or to terminate
2149 the use of a name identifier between itself and the identity provider.

2150

2151 3.6.1 Element `<ManageNameIDRequest>`

2152 A provider sends a `<ManageNameIDRequest>` message to inform the recipient of a changed name
2153 identifier or to indicate the termination of the use of a name identifier.

2154 The `<ManageNameIDRequest>` message SHOULD be signed or otherwise authenticated and integrity
2155 protected by the protocol binding used to deliver the message

2156 ~~The `<Issuer>` of the request MUST contain the unique identifier of the requesting provider, with a~~
2157 ~~Format value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.~~

2158 This message has the complex type **RegisterNameIDRequestType**, which extends
2159 **RequestAbstractType** and adds the following elements:

2160 `<NameID>` or `<EncryptedID>` [Required]

2161 The name identifier and associated attributes (in plaintext or encrypted form) that specify the principal
2162 as currently recognized by the identity and service providers prior to this request.

2163 `<NewID>` or `<NewEncryptedID>` or `<Terminate>` [Required]

2164 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2165 requesting provider concerning this principal or an indication that the use of the old identifier has been
2166 terminated. In the former case, if the requester is the service provider, the new identifier MUST
2167 appear in subsequent `<NameID>` elements in the `SPProvidedID` attribute. If the requester is the
2168 identity provider, the new value will appear in subsequent `<NameID>` elements as the element's value.

2169 The following schema fragment defines the `<ManageNameIDRequest>` element and its
2170 **ManageNameIDRequestType** complex type:

```
2171 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2172 <complexType name="ManageNameIDRequestType">
2173   <complexContent>
2174     <extension base="samlp:RequestAbstractType">
2175       <sequence>
2176         <choice>
2177           <element ref="saml:NameID"/>
2178           <element ref="saml:EncryptedID"/>
2179         </choice>

```

```

2180         <choice>
2181             <element ref="samlp:NewID"/>
2182             <element ref="samlp:NewEncryptedID"/>
2183             <element ref="samlp:Terminate"/>
2184         </choice>
2185     </sequence>
2186 </extension>
2187 </complexContent>
2188 </complexType>
2189 <element name="NewID" type="string"/>
2190 <element name="NewEncryptedID" type="saml:EncryptedIDType"/>
2191 <element name="Terminate" type="samlp:TerminateType"/>
2192 <complexType name="TerminateType">
2193     <sequence/>
2194 </complexType>

```

2195 3.6.2 Element <ManageNameIDResponse>

2196 The recipient of a <ManageNameIDRequest> message MUST respond with a
2197 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional
2198 content.

2199 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity
2200 protected by the protocol binding used to deliver the message.

2201 ~~The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a~~
2202 ~~Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.~~

2203 The following schema fragment defines the <ManageNameIDResponse> element:

```

2204 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>

```

2205 3.6.3 Processing Rules

2206 ~~The recipient MUST validate any signature present on the request or response message.~~

2207 If the request includes a <NameID> (or encrypted version) that the recipient does not recognize, the
2208 responding provider MUST respond with an error <Status> and MAY respond with a second-level
2209 <StatusCode> of ~~saml:urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal~~.

2210 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the
2211 case of a service provider) it will no longer accept assertions from the identifier provider or (in the case of
2212 an identity provider) it will no longer issue assertions to the service provider about the principal. The
2213 receiving provider MAY perform any maintenance with the knowledge that the relationship represented by
2214 the name identifier has been terminated. It MAY choose to invalidate the active session(s) of a principal
2215 for whom a relationship has been terminated.

2216 If the service provider requests that its identifier be changed by including a <NewID> (or
2217 <NewEncryptedID>) element, the identity provider MUST include the element's value as the
2218 SPProvidedID when subsequently communicating to the service provider regarding this principal.

2219 If the identity provider requests that its identifier be changed by including a <NewID> (or
2220 <NewEncryptedID>) element, the service provider MUST use the element's value as the <NameID>
2221 element value when subsequently communicating with the identity provider regarding this principal.

2222 In any case, the <NameID> value in the request and its associated SPProvidedID attribute MUST
2223 contain the most recent name identifier information established between the providers for the principal.

2224 In the case of an identifier with a Format Of `urn:oasis:names:tc:SAML:2.0:nameid-`
2225 `format:persistent`; Or `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted-opaque`
2226 `name-identifier`, the `NameQualifier` attribute MUST contain the unique identifier of the identity provider
2227 or be omitted.; If the identifier was established between the identity provider and an affiliation group of
2228 which the service provider is a member, then the `SPNameQualifier` attribute MUST contain the unique
2229 identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the service provider
2230 or be omitted.

2231 Changes to these identifiers may take a potentially significant amount of time to propagate through the
2232 systems at both the requester and the responder. Implementations might wish to allow each party to
2233 accept either identifier for some period of time following the successful completion of a name identifier
2234 change. Not doing so could result in the inability of the principal to access resources.

2235 All other processing rules associated with the underlying request and response messages MUST be
2236 observed.

2237 3.7 Single Logout Protocol

2238 The single logout protocol provides a message exchange protocol by which all sessions provided by a
2239 particular session authority are near-simultaneously terminated. The single logout protocol is used either
2240 when a principal logs out at a session participant or when the principal logs out directly at the
2241 session authority. This protocol may also be used to logout a principal due to a timeout. The reason for the
2242 logout event may be indicated through the `Reason` attribute.

2243 The principal may have established authenticated sessions both with the session authority, and individual
2244 session participants, based on authentication assertions supplied by the session authority.

2245 When the principal invokes the single logout process at a session participant, the session participant
2246 MUST send a `<LogoutRequest>` message to the session authority that provided the authentication
2247 `serviceassertion` related to that session at the session participant.-

2248 When either the principal invokes a logout at the session authority, or a session participant sends a logout
2249 request to the session authority specifying that principal, the session authority MUST send a
2250 `<LogoutRequest>` message to each session participant to which it provided authentication assertions
2251 under its current session with the principal, with the exception of the session participant that sent the
2252 `<LogoutRequest>` message to the session authority.-

2257 3.7.1 Element `<LogoutRequest>`

2258 A session participant or session authority sends a `<LogoutRequest>` message to indicate that a session
2259 has been terminated.

2260 The `<LogoutRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2261 by the protocol binding used to deliver the message.

2262 This message has the complex type `LogoutRequestType`, which extends `RequestAbstractType`, and
2263 adds the following elements and attributes:

2264 `<BaseID>` Or `<NameID>` or `<EncryptedID>` [Required]

2265 The `name-identifier` and associated attributes (in plaintext or encrypted form) that specify the principal
2266 as currently recognized by the identity and service providers prior to this request.

2267 `<SessionIndex>` [Optional]

2268 The identifier that indexes this session at the message recipient.

2269 NotOnOrAfter [Optional]

2270 The time at which the request expires.

2271 Reason [Optional]

2272 An indication of the reason for the logout, in the form of a URI reference.

2273 The following schema fragment defines the <LogoutRequest> element and associated
2274 **LogoutRequestType** complex type:

```
2275 <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
2276 <complexType name="LogoutRequestType">
2277   <complexContent>
2278     <extension base="samlp:RequestAbstractType">
2279       <sequence>
2280         <choice>
2281           <element ref="saml:BaseID"/>
2282           <element ref="saml:NameID"/>
2283           <element ref="saml:EncryptedID"/>
2284         </choice>
2285         <element ref="samlp:SessionIndex" minOccurs="0"
2286           minOccurs="unbounded"/>
2287       </sequence>
2288       <attribute name="Reason" type="anyURI" minOccurs="0"/>
2289       <attribute name="NotOnOrAfter" type="dateTime"
2290         minOccurs="0"/>
2291     </extension>
2292   </complexContent>
2293 </complexType>
2294 <element name="SessionIndex" type="string"/>
```

2295 3.7.2 Element <LogoutResponse>

2296 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of
2297 type **StatusResponseType**, with no additional content specified.

2298 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
2299 protected by the protocol binding used to deliver the message.

2300 The following schema fragment defines the <LogoutResponse> element:

```
2301 <element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

2302 3.7.3 Processing Rules

2303 ~~The <Issuer> of either message in this protocol MUST contain the unique identifier of the requesting or~~
2304 ~~responding provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-~~
2305 ~~format:entity.~~

2306 ~~Message recipients MUST validate any signature present on the messages specified in this protocol.~~

2307 The message sender MAY use the Reason attribute to indicate the reason for sending the
2308 <LogoutRequest>. Other values MAY be agreed upon between participants, but the following values
2309 are defined directly by this specification for use by all message senders:

2310 urn:oasis:names:tc:SAML:2.0:logout:user

2311 Specifies that the message is being sent because the principal wishes to terminate the indicated
2312 session.

2313 urn:oasis:names:tc:SAML:2.0:logout:admin

2314 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2315 session for that principal.

2316 All other processing rules associated with the underlying request and response messages MUST be
2317 observed.

2318 3.7.3.1 Session Participant Rules

2319 When a session participant receives a <LogoutRequest>, the session participant MUST authenticate
2320 the message. If the sender is the authority that provided an [authentication](#) assertion linked to the
2321 principal's current session, the session participant MUST invalidate the principal's session(s) referred to by
2322 the <BaseID>, <NameID>, or <EncryptedID> element, and any <SessionIndex> elements supplied
2323 in the message. [If no <SessionIndex> elements are supplied, then all sessions associated with the](#)
2324 [principal MUST be invalidated.](#)

2325
2326 The session participant MUST apply the logout request message to any assertion that meets the following
2327 conditions, even if the assertion arrives after the logout request:

- 2328 • The <SessionIndex> of one of the assertion's authentication statements matches one specified
2329 in the logout request, [or the logout request contains no <SessionIndex> elements.](#)
- 2330 • The assertion would otherwise be valid.
- 2331 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on
2332 the message).

2333 3.7.3.2 Session Authority Rules

2334 When a session authority receives a <LogoutRequest>, the session authority MUST authenticate the
2335 sender. If the sender is a session participant to which the session authority provided an [authentication](#)
2336 assertion for the current session, then the session authority SHOULD do the following [\(in the specified](#)
2337 [order\)](#):

- 2338 • [Send a <LogoutRequest> message to any session authority on behalf of whom the session](#)
2339 [authority proxied the user's authentication, unless the second authority is the originator of the](#)
2340 [<LogoutRequest>.](#)
- 2341 • Send a <LogoutRequest> message to each session participant for which the session authority
2342 provided assertions in the current session, *other than the originator of a current*
2343 *<LogoutRequest>.*
- 2344 • [Send a <LogoutRequest> message to any session authority on behalf of whom the session](#)
2345 [authority proxied the user's authentication, unless the second authority is the originator of the](#)
2346 [<LogoutRequest>.](#)
- 2347 • Terminate the principal's current session as specified by the <BaseID>, <NameID>, or
2348 <EncryptedID> element, and any <SessionIndex> elements present in the logout request
2349 message.-

2350 [If an error occurs during this further processing of the logout \(for example, other session participants may](#)
2351 [not all implement the particular single logout protocol binding used by the requesting session participant\),](#)
2352 [then the session authority MUST respond to the original requester with a <LogoutResponse> message,](#)
2353 [indicating the status of the logout request. The value](#)
2354 [urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding is provided for a second-level](#)
2355 [<StatusCode>, indicating that a session participant should retry the <LogoutRequest> using a](#)
2356 [different protocol binding.](#)

2357 | It should be noted that a session authority MAY initiate a logout for reasons other than having received a
 2358 | <LogoutRequest> from a session participant – these include, but are not limited to:

- 2359 | • If some timeout period was agreed out-of-band with an individual session participant, the session
 2360 | authority MAY send a <LogoutRequest> to that individual participant alone.
- 2361 | • An agreed global timeout period has been exceeded.
- 2362 | • The principal, or some other trusted entity has requested logout of the principal, directly at the
 2363 | session authority.
- 2364 | • The session authority has determined that the principal's credentials may have been compromised.

2365 | When constructing a logout request message, the session authority MUST set the value of the
 2366 | NotOnOrAfter attribute of the message to a time value, indicating an expiration time for the message.

2367 | In addition to the values specified in section 3.6.3 for the Reason attribute, the following values are also
 2368 | available for use by the session authority only:

2369 | urn:oasis:names:tc:SAML:2.0:logout:global-timeout

2370 | Specifies that the message is being sent because of the global session timeout interval period
 2371 | being exceeded.

2372 | urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2373 | Specifies that the message is being sent because a timeout interval period agreed between a
 2374 | participant and the authority has been exceeded.

2375 | ~~If an error occurs during this further processing of the logout (for example, relying session participants
 2376 | may not all implement the particular single logout protocol binding used by the requesting session
 2377 | participant), then the session authority MUST respond to the original requester with a
 2378 | <LogoutResponse> message, indicating the status of the logout request. The value
 2379 | samlp:UnsupportedBinding is provided for a second level <samlp:StatusCode>, indicating that a
 2380 | session participant should retry the <LogoutRequest> using a different protocol binding.~~

2381 | 3.8 Name Identifier Mapping Protocol

2382 | When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
 2383 | identifier for the same principal in a particular format or federation namespace, it can send a request to
 2384 | the identity provider using this protocol.

2385 | For example, a service provider that wishes to communicate with another service provider with whom it
 2386 | does not share an ~~identity federation identifier~~ for the principal can use an identity provider that shares an
 2387 | ~~identity federation identifier~~ for the principal with both service providers to map from its own ~~federated~~
 2388 | identifier to a new identifier, generally encrypted, with which it can communicate with the second service
 2389 | provider.

2390 | Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into an
 2391 | <EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

2392 | 3.8.1 Element <NameIDMappingRequest>

2393 | To request an alternate name identifier for a principal from an identity provider, a requester sends an
 2394 | <NameIDMappingRequest> message. This message has the complex type
 2395 | **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2396 <BaseID> or <NameID> or <EncryptedID> [Required]
2397 The identifier and associated attributes that specify the principal as currently recognized by the
2398 requester and the responder.

2399 <NameIDPolicy> [Required]

2400 The format and optional name qualifier that describes the requirements for the identifier to be
2401 returned.

2402 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2403 binding used to deliver the message.

2404 The following schema fragment defines the <NameIDMappingRequest> element and its
2405 **NameIDMappingRequestType** complex type:

```
2406 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>  
2407 <complexType name="NameIDMappingRequestType">  
2408   <complexContent>  
2409     <extension base="samlp:RequestAbstractType">  
2410       <sequence>  
2411         <choice>  
2412           <element ref="saml:BaseID"/>  
2413           <element ref="saml:NameID"/>  
2414           <element ref="saml:EncryptedID"/>  
2415         </choice>  
2416         <element ref="samlp:NameIDPolicy"/>  
2417       </sequence>  
2418     </extension>  
2419   </complexContent>  
2420 </complexType>
```

2421 3.8.2 Element <NameIDMappingResponse>

2422 The recipient of a <NameIDMappingRequest> message MUST respond with a
2423 <NameIDMappingResponse> message. This message has the complex type
2424 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following element:

2425 <NameID> or <EncryptedID> [Required]

2426 The identifier and associated attributes that specify the principal in the manner requested, usually in
2427 encrypted form.

2428 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2429 binding used to deliver the message.

2430 ~~The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a~~
2431 ~~Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.~~

2432 The following schema fragment defines the <NameIDMappingResponse> element and its
2433 **NameIDMappingResponseType** complex type:

```
2434 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>  
2435 <complexType name="NameIDMappingResponseType">  
2436   <complexContent>  
2437     <extension base="samlp:StatusResponseType">  
2438       <choice>  
2439         <element ref="saml:NameID"/>  
2440         <element ref="saml:EncryptedID"/>  
2441       </choice>  
2442     </extension>  
2443   </complexContent>
```

2444 </complexType>

2445 3.8.3 Processing Rules

2446 ~~The recipient MUST validate any signature present on the request or response message.~~

2447 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2448 <Status> containing a second-level <StatusCode> of
2449 ~~samlpurn:oasis:names:tc:SAML:2.0:status~~:UnknownPrincipal.

2450 At the responder's discretion, the
2451 ~~samlpurn:oasis:names:tc:SAML:2.0:status~~:InvalidNameIDPolicy status code MAY be
2452 returned to indicate an inability or unwillingness to supply an identifier in the requested format or
2453 namespace.

2454 All other processing rules associated with the underlying request and response messages MUST be
2455 observed.

2456 4 SAML Versioning

2457 The SAML specification set is versioned in two independent ways. Each is discussed in the following
2458 sections, along with processing rules for detecting and handling version differences, when applicable. Also
2459 included are guidelines on when and why specific version information is expected to change in future
2460 revisions of the specification.

2461 When version information is expressed as both a Major and Minor version, it may be expressed discretely,
2462 or in the form *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number
2463 $Major_A.Minor_A$ if and only if:

2464 $Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B > Minor_A)$

2465 4.1 SAML Specification Set Version

2466 Each release of the SAML specification set will contain a major and minor version designation describing
2467 its relationship to earlier and later versions of the specification set. The version will be expressed in the
2468 content and filenames of published materials, including the specification set document(s), and XML
2469 schema instance(s). There are no normative processing rules surrounding specification set versioning,
2470 since it merely encompasses the collective release of normative specification documents which
2471 themselves contain processing rules.

2472 The overall size and scope of changes to the specification set document(s) will informally dictate whether
2473 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards
2474 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics remain
2475 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
2476 revision.

2477 4.1.1 Schema Version

2478 As a non-normative documentation mechanism, any XML schema instances published as part of the
2479 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the
2480 specification set version in which it has been published. Validating implementations MAY use the attribute
2481 as a means of distinguishing which version of a schema is being used to validate messages, or to support
2482 a multiplicity of versions of the same logical schema.

2483 4.1.2 SAML Assertion Version

2484 The SAML `<Assertion>` element contains attributes for expressing the major and minor version of the
2485 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to
2486 document the syntax, semantics, and processing rules of the assertions of the same version. That is,
2487 specification set version 1.0 describes assertion version 1.0, and so on.

2488 There is explicitly NO relationship between the assertion version and the SAML assertion XML
2489 namespace that contains the schema definitions for that assertion version.

2490 The following processing rules apply:

- 2491 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported by
2492 the authority.
- 2493 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
2494 supported by the relying party.

- 2495 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version number
2496 is higher than the minor assertion version number supported by the relying party. However, all
2497 assertions that share a major assertion version number MUST share the same general processing
2498 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1
2499 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0
2500 assertion without ill effect.

2501 4.1.3 SAML Protocol Version

2502 The various SAML protocols' request and response elements contain attributes for expressing the major
2503 and minor version of the request or response message using a pair of integers. Each version of the SAML
2504 specification set will be construed so as to document the syntax, semantics, and processing rules of the
2505 protocol messages of the same version. That is, specification set version 1.0 describes request and
2506 response version V1.0, and so on.

2507 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace
2508 that contains the schema definitions for protocol messages for that protocol version.

2509 The version numbers used in SAML protocol request and response elements will be the same for any
2510 particular revision of the SAML specification set.

2511 4.1.3.1 Request Version

2512 The following processing rules apply to requests:

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

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 report
2532 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 top-
2534 level <StatusCode> value of [urn:oasis:names:tc:SAML:2.0:status:VersionMismatch](#), and

2535 MAY result in reporting one of the following second-level values:
2536 [urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh](#),
2537 [urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow](#), or
2538 [urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated](#).

2539 4.1.5 Permissible Version Combinations

2540 Assertions of a particular major version appear only in response messages of the same major version, as
2541 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2542 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2543 response message, if the appropriate assertion schema is referenced during namespace importation. But
2544 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major
2545 versions.

2546 4.2 SAML Namespace Version

2547 XML schema instances **and "qualified names" (QNames)** published as part of the specification set contain
2548 one or more target namespaces into which the type, element, and attribute definitions are placed. Each
2549 namespace is distinct from the others, and represents, in shorthand, the structural and syntactical
2550 definitions that make up that part of the specification.

2551 The namespace URIs defined by the specification set will generally contain version information of the form
2552 *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to the
2553 major and minor version of the specification set in which the namespace is first introduced and defined.
2554 This information is not typically consumed by an XML processor, which treats the namespace opaquely,
2555 but is intended to communicate the relationship between the specification set and the namespaces it
2556 defines.

2557 As a general rule, implementers can expect the namespaces (and the associated schema definitions)
2558 defined by a major revision of the specification set to remain valid and stable across minor revisions of the
2559 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but
2560 this is expected to be rare. In such cases, the older namespaces and their associated definitions should
2561 be expected to remain valid until a major specification set revision.

2562 4.2.1 Schema Evolution

2563 In general, maintaining namespace stability while adding or changing the content of a schema are
2564 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2565 older implementations will react to any given change, making forward compatibility difficult to achieve.
2566 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
2567 stability. Except in special circumstances (for example to correct major deficiencies or fix errors),
2568 implementations should expect forward compatible schema changes in minor revisions, allowing new
2569 messages to validate against older schemas.

2570 Implementations SHOULD expect and be prepared to deal with new extensions and message types in
2571 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types
2572 that leverage the extension facilities described in Section SAML Extensions. Older implementations
2573 SHOULD reject such extensions gracefully when they are encountered in contexts that dictate mandatory
2574 semantics. Examples include new query, statement, or condition types.

2575

5 SAML and XML Signature Syntax and Processing

2576 SAML assertions and SAML protocol request and response messages may be signed, with the following
2577 benefits:

- 2578 • An assertion signed by the SAML authority supports:
 - 2579 – Assertion integrity.
 - 2580 – Authentication of the SAML authority to a SAML relying party.
 - 2581 – If the signature is based on the SAML authority's public-private key pair, then it also provides for
2582 non-repudiation of origin.
- 2583 • A SAML protocol request or response message signed by the message originator supports:
 - 2584 – Message integrity.
 - 2585 – Authentication of message origin to a destination.
 - 2586 – If the signature is based on the originator's public-private key pair, then it also provides for non-
2587 repudiation of origin.

2588 A digital signature is not always required in SAML. For example, it may not be required in the following
2589 situations:

- 2590 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains
2591 protection from a signature on the containing protocol response message. "Inherited" signatures
2592 should be used with care when the contained object (such as the assertion) is intended to have a non-
2593 transitory lifetime. The reason is that the entire context must be retained to allow validation, exposing
2594 the XML content and adding potentially unnecessary overhead.
- 2595 • The SAML relying party or SAML requester may have obtained an assertion or protocol message from
2596 the SAML authority or SAML responder directly (with no intermediaries) through a secure channel,
2597 with the SAML authority or SAML responder having authenticated to the relying party or SAML
2598 responder by some means other than a digital signature.

2599 Many different techniques are available for "direct" authentication and secure channel establishment
2600 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
2601 the applicable security requirements depend on the communicating applications and the nature of the
2602 assertion or message transported.

2603 It is recommended that, in all other contexts, digital signatures be used for assertions and request and
2604 response messages. Specifically:

- 2605 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority
2606 SHOULD be signed by the SAML authority.
- 2607 • A SAML protocol message arriving at a destination from an entity other than the originating [sitesender](#)
2608 SHOULD be signed by the [senderorigin-site](#).

2609 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that
2610 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are
2611 intended to be the primary SAML signature mechanism, but the specification attempts to ensure
2612 compatibility with profiles that may require other mechanisms.

2613 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST
2614 be used if signing.

2615 5.1 Signing Assertions

2616 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema as
2617 described in Section Assertions.

2618 5.2 Request/Response Signing

2619 All SAML protocol request and response messages MAY be signed using the XML Signature. This is
2620 reflected in the schema as described in section 3.

2621 5.3 Signature Inheritance

2622 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
2623 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a
2624 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
2625 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,
2626 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
2627 interpretation should be equivalent to the case where the assertion itself was signed with the same key
2628 and signature options.

2629 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
2630 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may
2631 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
2632 information from the surrounding context, but no such inheritance should be inferred unless specifically
2633 identified by the profile.

2634 5.4 XML Signature Profile

2635 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
2636 and many choices. This section details the constraints on these facilities so that SAML processors do not
2637 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
2638 `xsd:ID`-typed attributes optionally present on the root elements to which signatures can apply, specifically
2639 the `ID` attribute on `<Assertion>` and the various request and response elements. These attributes are
2640 collectively referred to in this section as the identifier attributes.

2641 5.4.1 Signing Formats and Algorithms

2642 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
2643 detached.

2644 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
2645 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
2646 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

2647 5.4.2 References

2648 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the
2649 enclosing root element. The assertion's or protocol message's root element may or may not be the root
2650 element of the actual XML document containing the signed assertion or protocol message.

2651 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
2652 value of the root element of the message being signed. For example, if the attribute value is "foo", then
2653 the `URI` attribute in the `<ds:Reference>` element MUST be "#foo".

2654 **5.4.3 Canonicalization Method**

2655 SAML implementations SHOULD use Exclusive Canonicalization[Excl-C14N], with or without comments,
2656 both in the <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a
2657 <ds:Transform> algorithm. Use of Exclusive Canonicalization ensures that signatures created over
2658 SAML messages embedded in an XML context can be verified independent of that context.

2659 **5.4.4 Transforms**

2660 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
2661 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
2662 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
2663 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2664 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
2665 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can
2666 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
2667 applying the transforms manually to the content and reverifying the result as consisting of the same SAML
2668 message.

2669 **5.4.5 KeyInfo**

2670 XML Signature [XMLSig] defines usage of the <ds:KeyInfo> element. SAML does not require the
2671 use of <ds:KeyInfo> nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY
2672 be absent.

2673 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

2674 Use of signing does not affect [the](#) semantics of statements within assertions in any way, as stated in
2675 Section SAML Assertions.

2676 **5.4.7 Example**

2677 Following is an example of a signed response containing a signed assertion. Line breaks have been
2678 added for readability; the signatures are not valid and cannot be successfully verified.

2679 TODO: Update example.

```
2680 <Response  
2681   IssueInstant="2003-04-17T00:46:02Z"  
2682   MajorVersion="1"  
2683   MinorVersion="1"  
2684   Recipient="www.opensaml.org"  
2685   ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"  
2686   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
2687   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
2688   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2689   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
2690 <ds:Signature  
2691   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
2692 <ds:SignedInfo>  
2693 <ds:CanonicalizationMethod  
2694   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  
2695 <ds:SignatureMethod  
2696   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>  
2697 <ds:Reference  
2698   URI="#_c7055387-af61-4fce-8b98-e2927324b306">
```

```

2699 <ds:Transforms>
2700 <ds:Transform
2701   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
2702 <ds:Transform
2703   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2704 <InclusiveNamespaces
2705   PrefixList="#default saml samlp ds xsd xsi"
2706   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2707 </ds:Transform>
2708 </ds:Transforms>
2709 <ds:DigestMethod
2710   Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
2711 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
2712 </ds:Reference>
2713 </ds:SignedInfo>
2714 <ds:SignatureValue>
2715 x/GyPbzmfEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYwKf+5
2716 EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWaptaKlywS7gFgsD01qjyen3CP+m3D
2717 w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUae4=</ds:SignatureValue>
2718 <ds:KeyInfo>
2719 <ds:X509Data>
2720 <ds:X509Certificate>
2721 MIICyJCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaksCzAJBgNVBAYTA1VT
2722 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAQBGNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2723 F1VuaXZlcnNpdHkqb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
2724 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
2725 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
2726 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2727 Ym9yMQ4wDAYDVQQKEwVWQ0FJRDEcMBoGAlUEAxMTC2hpYjEuaW50ZXJ1ZXQyLmVh
2728 dTEhMCUGCSqGSIB3DQEJARYYcm9vdEBzaGlMS5pbnRlcml5dDIuZWRR1MIGfMA0G
2729 CSqGSIB3DQEBAAQAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
2730 IHRyQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIaOAPSZB113R6+KYiE7x4XAWIRCP+
2731 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
2732 pmqOIFGTWQIDAQABox0wGzAMBGNVHRMBAf8EAJAAMASGA1UdDwQEAwIFoDANBgkq
2733 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2734 qgi7lFV6MDkhhmTvtqBtjnmK3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
2735 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNRtREG8cCx3w/w==
2736 </ds:X509Certificate>
2737 </ds:X509Data>
2738 </ds:KeyInfo>
2739 </ds:Signature>
2740 <Status><StatusCode Value="samlp:Success"/></Status>
2741 <Assertion
2742   AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2743   IssueInstant="2003-04-17T00:46:02Z"
2744   Issuer="www.opensaml.org"
2745   MajorVersion="1"
2746   MinorVersion="1"
2747   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
2748   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2749   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2750 <Conditions
2751   NotBefore="2003-04-17T00:46:02Z"
2752   NotOnOrAfter="2003-04-17T00:51:02Z">
2753 <AudienceRestrictionCondition><Audience>http://www.opensaml.org</Audience>
2754 </AudienceRestrictionCondition></Conditions>
2755 <AuthenticationStatement
2756   AuthenticationInstant="2003-04-17T00:46:00Z"
2757   AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
2758 <Subject>
2759 <NameIdentifier
2760   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
2761 scott@example.org</NameIdentifier>
2762 <SubjectConfirmation>
2763 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>

```

```

2764 </SubjectConfirmation></Subject>
2765 <SubjectLocality
2766   IPAddress="127.0.0.1"/>
2767 </AuthenticationStatement>
2768 <ds:Signature
2769   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2770 <ds:SignedInfo>
2771 <ds:CanonicalizationMethod
2772   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2773 <ds:SignatureMethod
2774   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2775 <ds:Reference
2776   URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
2777 <ds:Transforms>
2778 <ds:Transform
2779   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
2780 <ds:Transform
2781   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2782 <InclusiveNamespaces
2783   PrefixList="#default saml samlp ds xsd xsi"
2784   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2785 </ds:Transform>
2786 </ds:Transforms>
2787 <ds:DigestMethod
2788   Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2789 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
2790 </ds:Reference>
2791 </ds:SignedInfo>
2792 <ds:SignatureValue>
2793 hq4zk+ZknjggCQgZm7ea8fI79gJESRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
2794 7iyziXBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtp3TD
2795 MWuL/cBUj2OtBZQQMFn7jQ9YB7klIz3RqVL+wNmeWI4=</ds:SignatureValue>
2796 <ds:KeyInfo>
2797 <ds:X509Data>
2798 <ds:X509Certificate>
2799 MIIICyJCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakkCzAJBgNVBAYTA1VT
2800 MRIWEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBACTB0lhZGlzb24xIDAeBgNVBAoT
2801 F1VuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLYyJEaXZpc2lvbiBvZiBJ
2802 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
2803 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoxZDA2MDkwNDA3Mjc1MVowgYsX
2804 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2805 Ym9yMQ4wDAYDVQQKEwVWVWQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJvZyLmVz
2806 dTEuMkUGCSqGSIB3DQEJARYYcm9vdEBzaGlMS5pbNlcm5ldDIuZW50ZG9uZG9u
2807 CSqGSIB3DQEBAQUAA4GNADCBiQKBggQDZSAb2svhAXnXVIVTx8vuRay+x50z7GJj
2808 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB1l3R6+KYiE7x4XAWIrcP+
2809 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6Dx8F8rvoP9W7027rhRjE
2810 pmqOIFGTWQIDAQABox0wGzAMBgNVHRMBAf8EajAAMASGA1UdDwQEAwIFoDANBgkq
2811 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujn/PizdN7s/z4D5d3pptWDJf2n
2812 qgi7lFV6MDkhnTvTqBtjmNk3No7v/dnP6Hr7wHxxCCRwubnmIfZ6QZAv2FU78pLX
2813 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGpdiowMNTREG8cCx3w/w==
2814 </ds:X509Certificate>
2815 </ds:X509Data>
2816 </ds:KeyInfo>
2817 </ds:Signature></Assertion></Response>

```

2818 6 SAML and XML Encryption Syntax and Processing

2819 Encryption is used as the means to implement confidentiality. The most common motives for
2820 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for
2821 competitive advantage or similar reasons. Confidentiality may also be required to insure the effectiveness
2822 of some other security mechanism. For example, a secret password or key may be encrypted.

2823 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 2824 • Communications confidentiality may be provided by mechanisms associated with a particular binding
2825 or profile. For example, the SOAP Binding ([SAMLBind]) supports the use of SSL/TLS or SOAP
2826 Message Security mechanisms for confidentiality.
- 2827 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`
2828 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be encrypted.
- 2829 • An entire assertion may be encrypted as described in section 2.4.4.
- 2830 • The `<BaseID>` or `<NameID>` element may be encrypted as described in section 2.3.3.
- 2831 • An `<Attribute>` element may be encrypted as described in section 2.5.3.2.

2832 6.1 General Considerations

2833 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use
2834 of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace
2835 the cleartext information in the same location within the XML instance. The `<EncryptedData>` element's
2836 `Type` attribute SHOULD be used and, if it is present, MUST have the value
2837 `http://www.w3.org/2001/04/xmlenc#Element`.

2838 Any of the algorithms defined for use with [XMLEnc] MAY be used to perform the encryption. The SAML
2839 schema is defined so that the inclusion of the encrypted data yields a valid instance.

2840 6.2 Combining Signatures and Encryption

2841 Use of [XMLEnc] and [XMLSig] MAY be combined. When an assertion is to be signed and encrypted the
2842 following rules apply. A relying party MUST perform signature validation and decryption in the reverse
2843 order that signing and encryption were performed.

- 2844 • When the entire assertion is encrypted, the signature MUST first be calculated and in place, and then
2845 the element encrypted.
- 2846 • When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be
2847 performed first and then the signature calculated over the assertion or message containing the
2848 encrypted element.

2849 6.3 Examples

2850 The following example shows an encrypted assertion in a response message:

2851 TBD

2852 The following example shows an encrypted name identifier.

2853 TBD

2854 7 SAML Extensions

2855 The SAML schemas support extensibility. An example of an application that extends SAML assertions is
2856 the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use
2857 the extensibility features in SAML to create extension schemas.

2858 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
2859 elements can serve as the head element of a substitution group. However, SAML types are not defined as
2860 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only
2861 elements and types that have been specifically designed to support extensibility.

2862 7.1 Assertion Schema Extension

2863 The SAML assertion schema is designed to permit separate processing of the assertion package and the
2864 statements it contains, if the extension mechanism is used for either part.

2865 The following elements are intended specifically for use as extension points in an extension schema; their
2866 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 2867 | • [<BaseID>](#)
- 2868 | • <Condition>
- 2869 | • <Statement>
- 2870 | • The following elements that are directly usable as part of SAML MAY be extended:
- 2871 | • <AuthnStatement>
- 2872 | • <AuthzDecisionStatement>
- 2873 | • <AttributeStatement>
- 2874 | • <AudienceRestriction>
- 2875 | • [<ProxyRestriction>](#)
- 2876 | • [<OneTimeUse>](#)

2877 The following elements are defined to allow elements from arbitrary namespaces within them, which
2878 serves as a built-in extension point without requiring an extension schema:

- 2879 | • [<BaseID>](#)
- 2880 | • <SubjectConfirmationData>
- 2881 | • <AttributeValue>
- 2882 | • <Advice>
- 2883 | • <AuthnContextDecl>

2884 7.2 Protocol Schema Extension

2885 The following SAML protocol elements are intended specifically for use as extension points in an
2886 extension schema; their types are set to `abstract`, and are thus usable only as the base of a derived
2887 type:

- 2888 • `<Query>`
- 2889 • `<SubjectQuery>`

2890 The following elements that are directly usable as part of SAML MAY be extended:

- 2891 • `<AuthnQuery>`
- 2892 • `<AuthzDecisionQuery>`
- 2893 • `<AttributeQuery>`

2894 8 SAML-Defined Identifiers

2895 The following sections define URI-based identifiers for common resource access actions, subject name
2896 identifier formats, and attribute name formats.

2897 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of
2898 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
2899 have one of the following stems:

```
2900 urn:oasis:names:tc:SAML:1.0:  
2901 urn:oasis:names:tc:SAML:1.1:  
2902 urn:oasis:names:tc:SAML:2.0:
```

2903 8.1 Action Namespace Identifiers

2904 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element ~~tion-Element~~
2905 `<Action>` (see ~~See~~ to refer to common sets of actions to perform on resources).

2906 8.1.1 Read/Write/Execute/Delete/Control

2907 **URI:** ~~urn:oasis:names:tc:SAML:1.0:~~<urn:oasis:names:tc:SAML:1.0:action:rwedc>

2908 Defined actions:

2909 Read Write Execute Delete Control

2910 These actions are interpreted as follows:

2911 Read

2912 The subject may read the resource.

2913 Write

2914 The subject may modify the resource.

2915 Execute

2916 The subject may execute the resource.

2917 Delete

2918 The subject may delete the resource.

2919 Control

2920 The subject may specify the access control policy for the resource.

2921 8.1.2 Read/Write/Execute/Delete/Control with Negation

2922 **URI:** ~~urn:oasis:names:tc:SAML:1.0:~~<urn:oasis:names:tc:SAML:1.0:action:rwedc-negation>

2923 Defined actions:

2924 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

2925 The actions specified in Section Read/Write/Execute/Delete/Control are interpreted in the same manner
2926 described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively
2927 specify that the stated permission is denied. Thus a subject described as being authorized to perform the
2928 action ~Read is affirmatively denied read permission.

2929 A SAML authority MUST NOT authorize both an action and its negated form.

2930 8.1.3 Get/Head/Put/Post

2931 | **URI:** <urn:oasis:names:tc:SAML:1.0:urn:oasis:names:tc:SAML:1.0:action:ghpp>

2932 Defined actions:

2933 GET HEAD PUT POST

2934 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
2935 the GET action on a resource is authorized to retrieve it.

2936 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST
2937 actions to the write permission. The correspondence is not exact however since an HTTP GET operation
2938 may cause data to be modified and a POST operation may cause modification to a resource other than
2939 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

2940 8.1.4 UNIX File Permissions

2941 | **URI:** <urn:oasis:names:tc:SAML:1.0:urn:oasis:names:tc:SAML:1.0:action:unix>

2942 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

2943 The action string is a four-digit numeric code:

2944 *extended user group world*

2945 Where the *extended* access permission has the value

2946 +2 if sgid is set

2947 +4 if suid is set

2948 The *user group* and *world* access permissions have the value

2949 +1 if execute permission is granted

2950 +2 if write permission is granted

2951 +4 if read permission is granted

2952 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read
2953 and execute; and world read.

2954 8.2 Attribute NameFormat Identifiers

2955 The following identifiers MAY be used in the NameFormat attribute defined on the
2956 | **AttributeDesignatorType** complex type (~~see Section x~~) to refer to the classification of the attribute name
2957 for purposes of interpreting the name.

2958 8.2.1 Unspecified

2959 | **URI:** <urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified>

2960 The interpretation of the attribute name is left to individual implementations.

2961 8.2.2 URI Reference

2962 **URI:** `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`

2963 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML
2964 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-
2965 specific. See ~~the Baseline Identities and Attributes specification [SAMLProf]~~ for a full discussion of
2966 representing names of XACML, X.500, and LDAP attributes~~attribute profiles that make use of this~~
2967 identifier.

2968 8.2.3 Basic

2969 **URI:** `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`

2970 The class of strings acceptable as the attribute name MUST be drawn from the set of values belonging to
2971 the primitive type **Name** as defined in Section 3.3.6 of [XML-Schema-Part2]. See [SAMLProf] for attribute
2972 profiles that make use of this identifier.

2973 8.3 NameID Format Identifiers

2974 The following identifiers MAY be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or
2975 `<Issuer>` elements (see Section 2.3) to refer to common formats for the content of the elements and the
2976 associated processing rules, if any.

2977 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of
2978 SAML.

2979 8.3.1 Unspecified

2980 **URI:** `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`

2981 The interpretation of the content of the element is left to individual implementations.

2982 8.3.2 Email Address

2983 **URI:** `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`

2984 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
2985 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that
2986 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in
2987 parentheses) after it, and is not surrounded by "<" and ">".

2988 8.3.3 X.509 Subject Name

2989 **URI:** `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`

2990 Indicates that the content of the element is in the form specified for the contents of the
2991 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors
2992 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
2993 differ from the rules given in IETF RFC 2253 [RFC 2253].

2994 **8.3.4 Windows Domain Qualified Name**

2995 | **URI: [urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName](#)**

2996 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
2997 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator
2998 MAY be omitted.

2999 **8.3.5 Kerberos Principal Name**

3000 | **URI: [urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos](#)**

3001 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3002 `name[/instance]@REALM`. The syntax, format and characters allowed for the name, instance, and
3003 realm are described in [RFC 1510].

3004 **8.3.6 Entity Identifier**

3005 | **URI: [urn:oasis:names:tc:SAML:2.0:nameid-format:entity](#)**

3006 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3007 (such as a SAML authority) or is a participant in SAML profiles (such as a service provider supporting the
3008 browser [SSO](#) profiles). Such an identifier can be used in the `<Issuer>` element to identify the issuer of a
3009 SAML request, response, or assertion, or within the `<NameID>` element to make assertions about system
3010 entities that can issue SAML requests, responses, and assertions. It can also be used in other elements
3011 and attributes whose purpose is to identify a system entity in various protocol exchanges.

3012 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3013 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3014 **8.3.7 Persistent Identifier**

3015 | **URI: [urn:oasis:names:tc:SAML:2.0:nameid-format:persistent](#)**

3016 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3017 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers
3018 generated by identity providers MUST be constructed using pseudo-random values that have no
3019 discernible correspondence with the subject's actual identifier (for example, username). The intent is to
3020 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.
3021 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3022 |

3023 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity
3024 provider that generated the identifier (see section 8.3.6). It MAY be omitted if the value can be derived
3025 from the context of the message containing the element, such as the issuer of an assertion.

3026 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service
3027 provider or affiliation of providers for whom the identifier was generated (see section 8.3.6). It MAY be
3028 omitted if the element is contained in a message intended only for consumption directly by the service
3029 provider, and the value would be the name of that service provider.

3030 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most
3031 recently set by the service provider or affiliation, if any (see section 3.6). If no such identifier has been
3032 established, than the attribute MUST be omitted.

3033 Persistent identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear text
3034 with providers other than the providers that have established the shared identifier. Furthermore, they
3035 MUST NOT appear in log files or similar locations without appropriate controls and protections.
3036 Deployments without such requirements are free to use other kinds of identifiers in their SAML
3037 exchanges, but MUST NOT overload this format with persistent but non-opaque values

3038 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3039 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3040 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3041 different and does not affect the behavior of the identity provider or any processing rules specific to
3042 persistent identifiers in the protocols defined in this specification.

3043 **8.3.8 Transient Identifier**

3044 | **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

3045 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3046 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3047 accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of
3048 256 characters.

3049 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3050 represents a transient and temporary pair-wise identifier, as described in Section 8.3.7. In such a case,
3051 | they MAY be omitted in accordance with the rules specified in that section.

3052 | 8.4

3053

9 References

3054 The following works are cited in the body of this specification.

3055 9.1 Normative References

- 3056 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web
3057 Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3058 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
3059 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.
3060 Note that this specification normatively references [Schema2], listed below.
- 3061 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium
3062 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3063 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World
3064 Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3065 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing,
3066 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web
3067 Consortium. Note that this specification normatively references [XMLEnc-XSD],
3068 listed below.
- 3069 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium.
3070 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3071 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January
3072 1999. <http://www.w3.org/TR/REC-xml-names>.
- 3073 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web
3074 Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this
3075 specification normatively references [XMLSig-XSD], listed below.
- 3076 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium.
3077 [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)
3078 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

3079 9.2 Non-Normative References

- 3080 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty
3081 Alliance Project, January 2003,
3082 [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)
3083 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3084 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of*
3085 *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December
3086 1978.
- 3087 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*.
3088 IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 3089 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*
3090 *Certificate and CRL Profile*. IETF RFC 2459, January 1999.
3091 <http://www.ietf.org/rfc/rfc2459.txt>.
- 3092 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF
3093 RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3094 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
3095 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

3096 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
3097 <http://www.ietf.org/rfc/rfc2246.txt>.

3098 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*
3099 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.
3100 <http://www.ietf.org/rfc/rfc2253.txt>.

3101 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF
3102 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3103 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.
3104 <http://www.ietf.org/rfc/rfc2630.txt>.

3105 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.
3106 <http://www.ietf.org/rfc/rfc2822.txt>.

3107 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,
3108 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3109 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF
3110 RFC 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3111 [SAMLAuthnCxt] J. Kemp. *Authentication Context for the OASIS Security Assertion Markup*
3112 *Language (SAML)*. OASIS, February 2004. Document ID sstc-saml-authn-
3113 context-2.0. <http://www.oasis-open.org/committees/security/>.

3114 [~~SAMLBaseAtts~~] ~~J. Hughes and P. Mishra. *Baseline Identities and Attributes for the OASIS*~~
3115 ~~*Security Assertion Markup Language (SAML)*. OASIS, March 2004. Document ID~~
3116 ~~sstc-saml-baseline-atts-2.0. <http://www.oasis-open.org/committees/security/>.~~

3117 [SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*
3118 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0.
3119 <http://www.oasis-open.org/committees/security/>.

3120 [SAMLProf] E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*
3121 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0.
3122 <http://www.oasis-open.org/committees/security/>.

3123 [~~SAMLMetadata~~] ~~E. Maler et al. *Metadata for the OASIS Security Assertion Markup Language*~~
3124 ~~*(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-metadata-2.0.~~
3125 ~~<http://www.oasis-open.org/committees/security/>.~~

3126 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*
3127 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID
3128 oasis-sstc-saml-conform-~~12.40~~. ~~HYPERLINK "[<http://www.oasis-open.org/committees/security/>.](http://www.oasis-
3129 open.org/committees/security/)~~

3131 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*
3132 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf)
3133 [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).

3134 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*
3135 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-~~42.40~~.
3136 ~~HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)~~
3137 ~~open.org/committees/security/"~~[http://www.oasis-](http://www.oasis-open.org/committees/security/)
[open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3138 [SAMLXSD] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID
3139 oasis-sstc-saml-schema-protocol-~~12.40~~. ~~HYPERLINK "[<http://www.oasis-open.org/committees/security/>.](http://www.oasis-
3140 open.org/committees/security/)~~

3142 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*
3143 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID
3144 oasis-sstc-saml-sec-consider-~~12.40~~. ~~HYPERLINK "[<http://www.oasis-open.org/committees/security/>.](http://www.oasis-
3145 open.org/committees/security/)~~

3144	[SAML-XSD]	E. Maler et al. <i>SAML assertion schema</i> . OASIS, September 2003. Document ID oasis-sstc-saml-schema-assertion-12.10. HYPERLINK " http://www.oasis-open.org/committees/security/ " http://www.oasis-open.org/committees/security/ .
3145		
3146		
3147	[SPKI]	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> . IETF RFC 2693, September 1999. http://www.ietf.org/rfc/rfc2693.txt .
3148		
3149		
3150	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
3151		
3152	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. http://www.w3.org/TR/WD-charreq .
3153		
3154	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0</i> . World Wide Web Consortium, April, 2002. http://www.w3.org/TR/charmod/ .
3155		
3156	[X.500]	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
3157		
3158	[XACML]	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml .
3159		
3160		
3161	[XKMS]	W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp. <i>XML Key Management Specification (XKMS)</i> . W3C Note 30 March 2001. http://www.w3.org/TR/xkms/ .
3162		
3163		

3164 **Appendix A. Acknowledgments**

3165 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
3166 Committee, whose voting members at the time of publication were:

- 3167 • @@

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/sstc-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/sstc-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/sstc-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/sstc-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/sstc-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. Addition of ECP header schema.

3169

3170

Appendix C. Notices

3171 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3172 might be claimed to pertain to the implementation or use of the technology described in this document or
3173 the extent to which any license under such rights might or might not be available; neither does it represent
3174 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
3175 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
3176 available for publication and any assurances of licenses to be made available, or the result of an attempt
3177 made to obtain a general license or permission for the use of such proprietary rights by implementors or
3178 users of this specification, can be obtained from the OASIS Executive Director.

3179 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
3180 other proprietary rights which may cover technology that may be required to implement this specification.
3181 Please address the information to the OASIS Executive Director.

3182 **Copyright © OASIS Open 2004. All Rights Reserved.**

3183 This document and translations of it may be copied and furnished to others, and derivative works that
3184 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
3185 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
3186 this paragraph are included on all such copies and derivative works. However, this document itself may
3187 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
3188 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
3189 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
3190 into languages other than English.

3191 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3192 or assigns.

3193 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3194 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3195 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3196 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.