



1

2

3

4

Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)

5

Document identifier: draft-sstc-core-21

6

Location: <http://www.oasis-open.org/committees/security/docs>

7

Publication date: 10 December 2001

8

Status: Interim draft. Send comments to the editors.

9

Contributors:[TBD]^[elm1]

10

Carlisle Adams, Entrust

11

Nigel Edwards, Hewlett-Packard

12

Marlena Erdos, Tivoli

13

Phillip Hallam-Baker, VeriSign, editor (pbaker@verisign.com)

14

Jeff Hodges, Oblix

15

Charles Knouse, Oblix

16

Chris McLaren, Netegrity

17

Prateek Mishra, Netegrity

18

RL "Bob" Morgan, University of Washington

19

Eve Maler, Sun Microsystems, editor (eve.maler@sun.com)

20

Tim Moses, Entrust

21

David Orchard, BEA

22

Irving Reid, Baltimore

23

23		
24	ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP	
25	LANGUAGE (SAML)	1
26	1. INTRODUCTION	5
27	1.1. NOTATION	5
28	1.2. SCHEMA ORGANIZATION AND NAMESPACES	5
29	1.3. SAML CONCEPTS (NON-NORMATIVE)	6
30	2. ASSERTIONS	7
31	2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	7
32	2.2. SIMPLE TYPES	7
33	2.2.1. <i>Simple Type IDType</i>	7
34	2.2.2. <i>Simple Type DecisionType</i>	8
35	2.3. ASSERTIONS	8
36	2.3.1. <i>Element <AssertionSpecifier></i>	8
37	2.3.2. <i>Element <AssertionID></i>	9
38	2.3.3. <i>Element <Assertion></i>	9
39	2.3.3.1. <i>Element <Conditions></i>	10
40	2.3.3.1.1. <i>Attributes NotBefore and NotOnOrAfter</i>	10
41	2.3.3.1.2. <i>Element <Condition></i>	11
42	2.3.3.1.3. <i>Element <AudienceRestrictionCondition></i>	11
43	2.3.3.2. <i>Element <Advice></i>	11
44	2.3.4. <i>Element <SingleAssertion></i>	12
45	2.3.5. <i>Element <MultipleAssertion></i>	12
46	2.4. STATEMENTS	13
47	2.4.1. <i>Element <Statement></i>	13
48	2.4.2. <i>Element <SubjectStatement></i>	13
49	2.4.2.1. <i>Element <Subject></i>	14
50	2.4.2.2. <i>Element <NameIdentifier></i>	14
51	2.4.2.3. <i>Element <SubjectConfirmation></i>	14

52	2.4.2.3.1	Element <ConfirmationMethod>	15
53	2.4.2.3.2	Element <SubjectConfirmationData>	15
54	2.4.3.	<i>Element <AuthenticationStatement></i>	15
55	2.4.3.1.	Element <AuthenticationLocality>	16
56	2.4.4.	<i>Element <AuthorizationStatement></i>	16
57	2.4.4.1.	Elements <Actions> and <Action>	17
58	2.4.4.2.	Element <Evidence>	17
59	2.4.5.	<i>Element <AttributeStatement></i>	17
60	2.4.5.1.	Elements <AttributeDesignator> and <Attribute>	18
61	2.4.5.1.1	Element <AttributeValue>	18
62	3.	PROTOCOL	19
63	3.1.	SCHEMA HEADER AND NAMESPACE DECLARATIONS	19
64	3.2.	SIMPLE TYPES	19
65	3.2.1.	<i>Simple Type CompletenessSpecifierType</i>	19
66	3.2.2.	<i>Simple Type StatusCodeType</i>	20
67	3.3.	REQUESTS	20
68	3.3.1.	<i>Complex Type RequestAbstractType</i>	20
69	3.3.2.	<i>Element <Request></i>	21
70	3.4.	QUERIES	21
71	3.4.1.	<i>Element <Query></i>	22
72	3.4.2.	<i>Element <SubjectQuery></i>	22
73	3.4.3.	<i>Element <AuthenticationQuery></i>	22
74	3.4.4.	<i>Element <AttributeQuery></i>	23
75	3.4.5.	<i>Element <AuthorizationQuery></i>	23
76	3.5.	RESPONSES	24
77	3.5.1.	<i>Complex Type ResponseAbstractType</i>	24
78	3.5.2.	<i>Element <Response></i>	24
79	4.	ASSERTION AND PROTOCOL VERSIONING	26

80	4.1. ASSERTION VERSION	26
81	4.2. REQUEST VERSION	26
82	4.3. RESPONSE VERSION	27
83	5. SCHEMA EXTENSION	28
84	5.1. ASSERTION SCHEMA EXTENSION	28
85	5.2. PROTOCOL SCHEMA EXTENSION	28
86	5.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	29
87	6. IDENTIFIERS	30
88	6.1. CONFIRMATION METHOD IDENTIFIERS	30
89	6.2. ACTION NAMESPACE IDENTIFIERS	30
90	7. SCHEMA LISTINGS	31
91	7.1. ASSERTION SCHEMA	31
92	7.2. PROTOCOL SCHEMA	34
93	8. REFERENCES	37
94	APPENDIX A. NOTICES	38
95		

96 1. Introduction

97 This specification defines the syntax and semantics for XML-encoded SAML assertions, protocol
98 requests, and protocol responses. These constructs are typically embedded in other structures
99 for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The SAML
100 specification for bindings and profiles **[SAMLBind]** provides frameworks for this embedding and
101 transport. Files containing just the SAML assertion schema **[SAML-XSD]** and protocol schema
102 **[SAML-P-XSD]** are available.

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

104 1.1. Notation

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

108 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
109 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be
110 interpreted as described in IETF RFC 2119 **[RFC2119]**:

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

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

117 Listings of SAML schemas appear like this.

118 Example code listings appear like this.

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

- 123 • The prefix `saml`: stands for the SAML assertion namespace.
- 124 • The prefix `samlp`: stands for the SAML request-response protocol namespace.
- 125 • The prefix `ds`: stands for the W3C XML Signature namespace.
- 126 • The prefix `xsd`: stands for the W3C XML Schema namespace in example listings. In
127 schema listings, this is the default namespace and no prefix is shown.

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

130 1.2. Schema Organization and Namespaces

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

133 <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd>

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

136 <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd>

137 **Note:** The SAML namespace names are temporary and will change when
138 SAML 1.0 is finalized.

139 The assertion schema is imported into the protocol schema. Also imported into both schemas is
140 the schema for XML Signature **[XMLSig-XSD]**, which is associated with the following XML
141 namespace^[elm2]:

142 <http://www.w3.org/2000/09/xmldsig#>

143 The XML Signature element `<ds:KeyInfo>`, defined in **[XMLSig]** §4.4, is of particular interest in
144 SAML.

145 **1.3. SAML Concepts (Non-Normative)**

146 This section is informative only and is superseded by any contradicting information in the
147 normative text in Sections 1.2 and following. A glossary of SAML terms and concepts
148 **[SAMLGloss]** is available.

149 [TBD]^[elm3]

150 2. Assertions

151 An assertion is a package of information that supplies one or more statements made by an
152 issuer. SAML allows issuers to make three different kinds of assertion statement:

- 153 • **Authentication:** The specified subject was authenticated by a particular means at a
154 particular time.
- 155 • **Authorization decision:** A request to allow the specified subject to access the specified
156 object has been granted or denied.
- 157 • **Attribute:** The specified subject is associated with the supplied attributes.

158 Assertions have a nested structure. A series of inner elements representing authentication
159 statements, authorization decision statements, and attribute statements contains the specifics,
160 while an outer generic assertion element provides information that is common to all the
161 statements.

162 2.1. Schema Header and Namespace Declarations

163 The following schema fragment defines the XML namespaces and other header information for
164 the assertion schema:

```
165 <schema  
166   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-  
167   sstc-schema-assertion-21.xsd"  
168   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
169   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
170   schema-assertion-21.xsd"  
171   xmlns="http://www.w3.org/2001/XMLSchema"  
172   elementFormDefault="unqualified">  
173   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
174     schemaLocation="xmldsig-core-schema.xsd"/>  
175   <annotation>  
176     <documentation>draft-sstc-schema-assertion-21.xsd</documentation>  
177   </annotation>  
178   ...  
179 </schema>
```

180 2.2. Simple Types

181 The following sections define the SAML assertion-related simple types.

182 2.2.1. Simple Type IDType

183 The **IDType** simple type is used to declare and reference identifiers to assertions, requests and
184 responses.

185 Values of attributes declared to be of type **IDType** MUST satisfy the following properties:

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

190 The mechanism by which the application ensures that the identifier is unique is left to the
191 implementation. In the case that a pseudorandom technique is employed the probability of two
192 randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less than 2^{-160} .
193

194 It is OPTIONAL for an identifier based on **IDType** to be resolvable in principle to some resource.
195 In the case that the identifier is resolvable in principle (for example, the identifier is in the form of
196 a URI reference), it is OPTIONAL for the identifier to be dereferenceable.

197 The following schema fragment defines the **IDType** type:

```
198 <simpleType name="IDType">  
199   <restriction base="string"/>  
200 </simpleType>
```

201 2.2.2. Simple Type DecisionType

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

204 Permit

205 The specified action is permitted.

206 Deny

207 The specified action is denied.

208 Indeterminate

209 No assessment is made as to whether the specified action is permitted or denied.

210 The following schema fragment defines the **DecisionType** type:

```
211 <simpleType name="DecisionType">  
212   <restriction base="string">  
213     <enumeration value="Permit"/>  
214     <enumeration value="Deny"/>  
215     <enumeration value="Indeterminate"/>  
216   </restriction>  
217 </simpleType>
```

218 2.3. Assertions

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

220 2.3.1. Element <AssertionSpecifier>

221 The <AssertionSpecifier> element specifies an assertion either by reference or by value. It
222 contains one of the following subelements:

223 <AssertionID>

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

225 <Assertion>

226 Specifies an assertion by value.

227 <SingleAssertion>

228 Specifies an assertion containing a single statement by value.

229 <MultipleAssertion>

230 Specifies an assertion containing multiple statements by value.

231 The following schema fragment defines the <AssertionSpecifier> element and its
232 **AssertionSpecifierType** complex type:

```
233 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>  
234 <complexType name="AssertionSpecifierType">  
235   <choice>  
236     <element ref="saml:AssertionID"/>  
237     <element ref="saml:Assertion"/>  
238     <element ref="saml:SingleAssertion"/>  
239     <element ref="saml:MultipleAssertion"/>
```


240 `</choice>`
241 `</complexType>`

242 2.3.2. Element <AssertionID>

243 The <AssertionID> element specifies a reference to the identifier of a SAML assertion.

244 The following schema fragment defines the <AssertionID> element and **IDType** type:

245 `<element name="AssertionID" type="saml:IDType"/>`

246 2.3.3. Element <Assertion>

247 A SAML assertion is specified by an XML element whose type is derived from the abstract XML
248 type **AssertionAbstractType**. This type specifies the basic information that is common to all
249 assertions. Instances of SAML assertions have concrete types that are extensions of the base
250 **AssertionAbstractType**.

251 The element <Assertion> of abstract type **AssertionAbstractType** is used to specify a SAML
252 assertion:

253 **MajorVersion** [Required]

254 Each assertion MUST specify the SAML major version identifier as an integer in this attribute.

255 The identifier for this version of SAML is 1. Processing of this attribute is specified in Section 4.

256 **MinorVersion** [Required]

257 Each assertion MUST specify the SAML minor version identifier as an integer in this attribute.

258 The identifier for this version of SAML is 0. Processing of this attribute is specified in Section 4.

259 **AssertionID** [Required]

260 The identifier for this assertion.

261 **Issuer** [Required]

262 The issuer of the assertion. The name of the issuer is provided as a string[PHB4].

263 **IssueInstant** [Required]

264 The time instant of issue.

265 **<Conditions>** [Optional]

266 Conditions that MUST be taken into account in assessing the validity of the assertion.

267 **<Advice>** [Optional]

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

270 The following schema fragment defines the <Assertion> element and its

271 **AssertionAbstractType** complex type:

272 `<element name="Assertion" type="saml:AssertionAbstractType"/>`
273 `<complexType name="AssertionAbstractType" abstract="true">`
274 `<sequence>`
275 `<element ref="saml:Conditions" minOccurs="0"/>`
276 `<element ref="saml:Advice" minOccurs="0"/>`
277 `</sequence>`
278 `<attribute name="MajorVersion" type="integer" use="required"/>`
279 `<attribute name="MinorVersion" type="integer" use="required"/>`
280 `<attribute name="AssertionID" type="saml:IDType" use="required"/>`
281 `<attribute name="Issuer" type="string" use="required"/>`
282 `<attribute name="IssueInstant" type="dateTime" use="required"/>`
283 `</complexType>`

284 2.3.3.1. Element <Conditions>

285 If an assertion contains a <Conditions> element, the validity of the assertion is dependent on
286 the conditions provided. Each condition evaluates to a status of Valid, Invalid, or
287 Indeterminate. The validity status of an assertion is the conjunction of the validity of each of
288 the conditions it contains, as follows:

- 289 • If any condition evaluates to Invalid, the assertion status is Invalid.
- 290 • If no condition evaluates to Invalid and one or more conditions evaluate to
291 Indeterminate, the assertion status is Indeterminate.
- 292 • If no conditions are specified or all the specified conditions evaluate to Valid, the
293 assertion status is Valid.

294 The <Conditions> element MAY be extended to define additional conditions. If an element
295 contained within a <Conditions> element is encountered that is not understood, the status of
296 the condition MUST be evaluated to Indeterminate.

297 The <Conditions> element contains the following element and attributes:

298 NotBefore [Optional]

299 Specifies the earliest time instant at which the assertion is valid.

300 NotOnOrAfter [Optional]

301 Specifies the time instant at which the assertion has expired.

302 <Condition> [Zero or more]

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

304 <AudienceRestrictionCondition> [Any Number]

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

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

```
308 <element name="Conditions" type="saml:ConditionsType"/>  
309 <complexType name="ConditionsType">  
310 <choice minOccurs="0" maxOccurs="unbounded">  
311 <element ref="saml:Condition"/>  
312 <element ref="saml:AudienceRestrictionCondition"/>  
313 </choice>  
314 <attribute name="NotBefore" type="dateTime" use="optional"/>  
315 <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
316 </complexType>
```

317 2.3.3.1.1 Attributes NotBefore and NotOnOrAfter

318 The NotBefore and NotOnOrAfter attributes specify time limits on the validity of the assertion.

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

321 If the value for either NotBefore or NotOnOrAfter is omitted or is equal to the start of the
322 epoch it is considered unspecified. If the NotBefore attribute is unspecified (and if any other
323 conditions that are supplied evaluate to Valid), the assertion is valid at any time before the time
324 instant specified by the NotOnOrAfter attribute. If the NotOnOrAfter attribute is unspecified
325 (and if any other conditions that are supplied evaluate to Valid), the assertion is valid from the
326 time instant specified by the NotBefore attribute with no expiry. If neither attribute is specified
327 (and if any other conditions that are supplied evaluate to Valid), the assertion is valid at any
328 time.

329 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type
330 that is built in to the W3C XML Schema Datatypes specification [**Schema2**]. All time instants are
331 interpreted to be in Universal Coordinated Time (UTC) unless they explicitly indicate a time zone.
332 Implementations MUST NOT generate time instants that specify leap seconds.

333 **2.3.3.1.2 Element <Condition>**

334 The `<Condition>` element serves as an extension point for new conditions.

335 The following schema fragment defines the `<Condition>` element and its
336 **ConditionAbstractType** complex type:

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

339 **2.3.3.1.3 Element <AudienceRestrictionCondition>**

340 The `<AudienceRestrictionCondition>` element specifies that the assertion is addressed to
341 one or more specific audiences. Although a party that is outside the audiences specified is
342 capable of drawing conclusions from an assertion, the issuer explicitly makes no representation
343 as to accuracy or trustworthiness to such a party.

344 An audience is identified by a URI. The URI MAY identify a document that describes the terms
345 and conditions of audience membership.

346 The condition evaluates to `Valid` if and only if the relying party is a member of one or more of
347 the audiences specified.

348 The issuer of an assertion cannot prevent a party to whom it is disclosed making a decision on
349 the basis of the information provided. However, the `<AudienceRestrictionCondition>`
350 element allows the issuer to state explicitly that no warranty is provided to such a party in a
351 machine- and human-readable form. While there can be no guarantee that a court would
352 upholding such a warranty exclusion in every circumstance, the probability of upholding the
353 warranty exclusion is considerably improved.

354 The following schema fragment defines the `<AudienceRestrictionCondition>` element and
355 its **AudienceRestrictionConditionType** complex type:

```
356 <element name="AudienceRestrictionCondition"  
357     type="saml:AudienceRestrictionConditionType"/>  
358 <complexType name="AudienceRestrictionConditionType">  
359     <complexContent>  
360         <extension base="saml:ConditionAbstractType">  
361             <sequence>  
362                 <element ref="saml:Audience"  
363                     minOccurs="1" maxOccurs="unbounded"/>  
364             </sequence>  
365         </extension>  
366     </complexContent>  
367 </complexType>  
368 <element name="Audience" type="anyURI"/>
```

369 **2.3.3.2. Element <Advice>**

370 The `<Advice>` element contains any additional information that the issuer wishes to provide.
371 This information MAY be ignored by applications without affecting either the semantics or the
372 validity of the assertion. The `<Advice>` element serves as an extension point for specialized
373 kinds of advice.

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

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

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

```
381 <element name="Advice" type="saml:AdviceType"/>
382 <complexType name="AdviceType">
383   <sequence>
384     <choice minOccurs="0" maxOccurs="unbounded">
385       <element ref="saml:AssertionSpecifier"/>
386       <element ref="saml:AdviceElement"/>
387       <any namespace="##other" processContents="lax"/>
388     </choice>
389   </sequence>
390 </complexType>
391 <element name="AdviceElement" type="saml:AdviceAbstractType"/>
392 <complexType name="AdviceAbstractType"/>
```

393 2.3.4. Element <SingleAssertion>

394 The <SingleAssertion> element specifies a single statement. It contains one of the following:

395 <Statement>

396 A statement defined in an extension schema.

397 <SubjectStatement>

398 A subject statement defined in an extension schema.

399 <AuthenticationStatement>

400 An authentication statement.

401 <AuthorizationStatement>

402 An authorization decision statement.

403 <AttributeStatement>

404 An attribute statement.

405 The following schema fragment defines the <SingleAssertion> element and its
406 **SingleAssertionType** complex type:

```
407 <element name="SingleAssertion" type="saml:SingleAssertionType"/>
408 <complexType name="SingleAssertionType">
409   <complexContent>
410     <extension base="saml:AssertionAbstractType">
411       <choice>
412         <element ref="saml:Statement"/>
413         <element ref="saml:SubjectStatement"/>
414         <element ref="saml:AuthenticationStatement"/>
415         <element ref="saml:AuthorizationStatement"/>
416         <element ref="saml:AttributeStatement"/>
417       </choice>
418     </extension>
419   </complexContent>
420 </complexType>
```

421 2.3.5. Element <MultipleAssertion>

422 The <MultipleAssertion> element specifies a series of zero or more statements. It can
423 contain the same statement elements that are allowed in <SingleAssertion>. Multiple

424 statements in a <MultipleAssertion> element MUST be interpreted identically to several
425 <SingleAssertion> elements with the same common information that contain the statements
426 individually.

427 The following schema fragment defines the <MultipleAssertion> element and its
428 **MultipleAssertionType** complex type:

```
429 <element name="MultipleAssertion" type="saml:MultipleAssertionType"/>  
430 <complexType name="MultipleAssertionType">  
431 <complexContent>  
432 <extension base="saml:AssertionAbstractType">  
433 <choice minOccurs="0" maxOccurs="unbounded">  
434 <element ref="saml:Statement"/>  
435 <element ref="saml:SubjectStatement"/>  
436 <element ref="saml:AuthenticationStatement"/>  
437 <element ref="saml:AuthorizationStatement"/>  
438 <element ref="saml:AttributeStatement"/>  
439 </choice>  
440 </extension>  
441 </complexContent>  
442 </complexType>
```

443 2.4. Statements

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

445 2.4.1. Element <Statement>

446 The <Statement> element is an extension point that allows other assertion-based applications
447 to reuse the SAML assertion framework. Its **StatementAbstractType** complex type is abstract;
448 extension elements MUST use the `xsi:type` attribute to indicate the derived type.

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

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

453 2.4.2. Element <SubjectStatement>

454 The <SubjectStatement> element is an extension point that allows other assertion-based
455 applications to reuse the SAML assertion framework. It contains a <Subject> element that
456 allows an issuer to describe a subject. Its **SubjectStatementAbstractType** complex type, which
457 extends **StatementAbstractType**, is abstract; extension elements MUST use the `xsi:type`
458 attribute to indicate the derived type.

459 The following schema fragment defines the <SubjectAssertion> element and its
460 **SubjectAssertionAbstractType** abstract type:

```
461 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
462 <complexType name="SubjectStatementAbstractType" abstract="true">  
463 <complexContent>  
464 <extension base="saml:StatementAbstractType">  
465 <sequence>  
466 <element ref="saml:Subject"/>  
467 </sequence>  
468 </extension>  
469 </complexContent>  
470 </complexType>
```

471 **2.4.2.1. Element <Subject>**

472 The <Subject> element specifies one or more subjects. It contains a mixture of one or more of
473 the following elements:

474 <NameIdentifier>

475 An identification of a subject by its name and security domain.

476 <SubjectConfirmation>

477 Information that allows the subject to be authenticated.

478 <AssertionSpecifier>

479 An identification of a subject by reference to or containment of an assertion.

480 If a <Subject> element contains more than one subject specification, the issuer is asserting that
481 the surrounding statement is true for all of the subjects specified. For example, if both a
482 <NameIdentifier> and a <SubjectConfirmation> element are present, the issuer is
483 asserting that the statement is true of both subjects being identified. A <Subject> element
484 SHOULD NOT identify more than one principal.

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

```
487 <element name="Subject" type="saml:SubjectType"/>  
488 <complexType name="SubjectType">  
489 <choice maxOccurs="unbounded">  
490 <element ref="saml:NameIdentifier"/>  
491 <element ref="saml:SubjectConfirmation"/>  
492 <element ref="saml:AssertionSpecifier"/>  
493 </choice>  
494 </complexType>
```

495 **2.4.2.2. Element <NameIdentifier>**

496 The <NameIdentifier> element specifies a subject by a combination of a name and a security
497 domain. It has the following attributes:

498 SecurityDomain

499 The security domain governing the name of the subject.

500 Name

501 The name of the subject.

502 The interpretation of the security domain and the name are left to individual implementations,
503 including issues of anonymity, pseudonymity, and the persistence of the identifier with respect to
504 the asserting and relying parties.

505 The following schema fragment defines the <NameIdentifier> element and its
506 **NameIdentifierType** complex type:

```
507 <element name="NameIdentifier" type="saml:NameIdentifierType"/>  
508 <complexType name="NameIdentifierType">  
509 <attribute name="SecurityDomain" type="string"/>  
510 <attribute name="Name" type="string"/>  
511 </complexType>
```

512 **2.4.2.3. Element <SubjectConfirmation>**

513 The <SubjectConfirmation> element specifies a subject by supplying data that allows the
514 subject to be authenticated. It contains the following elements in order:

515 <ConfirmationMethod> [One or more]

516 A URI that identifies a protocol to be used to authenticate the subject. URIs identifying common
517 authentication protocols are specified in Section 6.

518 <SubjectConfirmationData> [Zero or more]
519 Additional authentication information to be used by a specific authentication protocol.
520 <ds:KeyInfo> [Optional]
521 An XML Signature [XMLSig] element that specifies a cryptographic key held by the subject.

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

```
524 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
525 <complexType name="SubjectConfirmationType">  
526 <sequence>  
527 <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>  
528 <element name="SubjectConfirmationData" type="string" minOccurs="0"/>  
529 <element ref="ds:KeyInfo" minOccurs="0"/>  
530 </sequence>  
531 <!-- Need to modify this element-->  
532 </complexType>
```

533 2.4.2.3.1 Element <ConfirmationMethod>

534 The following schema fragment defines the <ConfirmationMethod> element:

```
535 <element name="ConfirmationMethod" type="anyURI"/>
```

536 2.4.2.3.2 Element <SubjectConfirmationData>

537 The following schema fragment defines the <SubjectConfirmationData> element:

```
538 <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
```

539 2.4.3. Element <AuthenticationStatement>

540 The <AuthenticationStatement> element supplies a statement by the issuer that its subject
541 was authenticated by a particular means at a particular time. It is of type
542 **AuthenticationStatementType**, which extends **SubjectStatementAbstractType** with the
543 addition of the following element and attributes:

544 AuthenticationMethod [Required]
545 Specifies the type of authentication that took place.

546 AuthenticationInstant [Required]
547 Specifies the time at which the authentication took place.

548 <AuthenticationLocality> [Optional]
549 Specifies the DNS domain name and IP address for the system entity that performed the
550 authentication.

551 The following schema fragment defines the <AuthenticationStatement> element and its
552 **AuthenticationStatementType** complex type:

```
553 <element name="AuthenticationStatement"  
554 <complexType name="AuthenticationStatementType">  
555 <complexContent>  
556 <extension base="saml:SubjectStatementAbstractType">  
557 <sequence>  
558 <element ref="saml:AuthenticationLocality" minOccurs="0"/>  
559 </sequence>  
560 <attribute name="AuthenticationMethod" type="anyURI"/>  
561 <attribute name="AuthenticationInstant" type="dateTime"/>  
562 </extension>  
563 </complexContent>  
564 </complexType>
```

566 2.4.3.1. Element <AuthenticationLocality>

567 The <AuthenticationLocality> element specifies the DNS domain name and IP address
568 for the system entity that was authenticated. It has the following attributes:

569 IPAddress [Optional]

570 The IP address of the system entity that that was authenticated.

571 DNSAddress [Required]

572 The DNS address of the system entity that that was authenticated.

573 This element is entirely advisory, since both these fields are quite easily “spoofed” but current
574 practice appears to require its inclusion.

575 The following schema fragment defines the <AuthenticationLocality> element and its
576 **AuthenticationLocalityType** complex type:

```
577 <element name="AuthenticationLocality"  
578         type="saml:AuthenticationLocalityType"/>  
579 <complexType name="AuthenticationLocalityType">  
580   <attribute name="IPAddress" type="string" use="optional"/>  
581   <attribute name="DNSAddress" type="string" use="optional"/>  
582 </complexType>
```

583 2.4.4. Element <AuthorizationStatement>

584 The <AuthorizationStatement> element supplies a statement by the issuer that the request
585 for access by the specified subject to the specified resource has resulted in the specified decision
586 on the basis of some optionally specified evidence. It is of type **AuthorizationStatementType**,
587 which extends **SubjectStatementAbstractType** with the addition of the following elements (in
588 order) and attributes:

589 Resource [Optional]

590 A URI identifying the resource to which access authorization is sought.

591 Decision [Optional]

592 The decision rendered by the issuer with respect to the specified object. The value is of the
593 **DecisionType** simple type.

594 <Actions> [Required]

595 The set of actions authorized for the specified resource.

596 <Evidence> [Zero or more]

597 A set of assertions that the issuer relied on in making the decision.

598 The following schema fragment defines the <AuthorizationStatement> element and its
599 **AuthorizationStatementType** complex type:

```
600 <element name="AuthorizationStatement"  
601         type="saml:AuthorizationStatementType"/>  
602 <complexType name="AuthorizationStatementType">  
603   <complexContent>  
604     <extension base="saml:SubjectStatementAbstractType">  
605       <sequence>  
606         <element ref="saml:Actions"/>  
607         <element ref="saml:Evidence"  
608           minOccurs="0" maxOccurs="unbounded"/>  
609       </sequence>  
610       <attribute name="Resource" type="anyURI" use="optional"/>  
611       <attribute name="Decision"  
612         type="saml:DecisionType" use="optional"/>  
613     </extension>  
614   </complexContent>  
615 </complexType>
```


616 2.4.4.1. Elements <Actions> and <Action>

617 The <Actions> element specifies the set of actions on the specified resource for which
618 permission is sought. It has the following element and attribute:

619 Namespace [Optional]

620 A URI representing the namespace in which the names of specified actions are to be interpreted.

621 If this element is absent, the namespace specified in section [TBD]_[elm5] is in effect by default.

622 <Action> [One or more]

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

624 The following schema fragment defines the <Actions> element, its **ActionTypes** complex type,
625 and the <Action> element:

```
626 <element name="Actions" type="saml:ActionTypes"/>  
627 <complexType name="ActionTypes">  
628 <sequence>  
629 <element ref="saml:Action" maxOccurs="unbounded"/>  
630 </sequence>  
631 <attribute name="Namespace" type="anyURI" use="optional"/>  
632 </complexType>  
633 <element name="Action" type="string"/>
```

634 2.4.4.2. Element <Evidence>

635 The <Evidence> element contains an assertion that the issuer relied on in issuing the
636 authorization decision. It has the **AssertionSpecifierType** complex type.

637 The provision of an assertion as evidence MAY affect the reliance agreement between the client
638 and the service. For example, in the case that the client presented an assertion to the service in a
639 request, the service MAY use that assertion as evidence in making its response without
640 endorsing the assertion as valid either to the client or any third party.

641 The following schema fragment defines the <Evidence> element:

```
642 <element name="Evidence" type="saml:AssertionSpecifierType"/>
```

643 2.4.5. Element <AttributeStatement>

644 The <AttributeStatement> element supplies a statement by the issuer that the specified
645 subject is associated with the specified attributes. It is of type **AttributeStatementType**, which
646 extends **SubjectStatementAbstractType** with the addition of the following element:

647 <Attribute> [One or More]

648 The <Attribute> element specifies an attribute of the subject.

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

```
651 <element name="AttributeStatement" type="saml:AttributeStatementType"/>  
652 <complexType name="AttributeStatementType">  
653 <complexContent>  
654 <extension base="saml:SubjectStatementAbstractType">  
655 <sequence>  
656 <element ref="saml:Attribute" maxOccurs="unbounded"/>  
657 </sequence>  
658 </extension>  
659 </complexContent>  
660 </complexType>
```

661 **2.4.5.1. Elements <AttributeDesignator> and <Attribute>**

662 The <AttributeDesignator> element identifies an attribute name within an attribute
663 namespace. It has the **AttributeDesignatorType** complex type. It is used in an attribute
664 assertion query to request that attribute values within a specific namespace be returned. The
665 <AttributeDesignator> element contains the following attributes:

666 AttributeNamespace [Required]

667 The AttributeNamespace attribute specifies the namespace in which the AttributeName
668 elements are interpreted.

669 AttributeName [Required]

670 The AttributeName attribute specifies the name of the attribute.

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

```
673 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
674 <complexType name="AttributeDesignatorType">  
675 <attribute name="AttributeName" type="string"/>  
676 <attribute name="AttributeNamespace" type="anyURI"/>  
677 </complexType>
```

678 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the
679 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the
680 following element:

681 <AttributeValue> [Required]

682 The value of the attribute.

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

```
685 <element name="Attribute" type="saml:AttributeType"/>  
686 <complexType name="AttributeType">  
687 <complexContent>  
688 <extension base="saml:AttributeDesignatorType">  
689 <sequence>  
690 <element ref="saml:AttributeValue"/>  
691 </sequence>  
692 </extension>  
693 </complexContent>  
694 </complexType>
```

695 **2.4.5.1.1 Element <AttributeValue>**

696 The <AttributeValue> element supplies the value of the specified attribute. It is of the
697 **AttributeValueType** complex type, which allows the inclusion of any element in any namespace
698 and specifies that lax schema validation is in effect.

699 The following schema fragment defines the <AttributeValue> element and its
700 **AttributeValueType** complex type:

```
701 <element name="AttributeValue" type="saml:AttributeValueType"/>  
702 <complexType name="AttributeValueType">  
703 <sequence>  
704 <any namespace="##any" processContents="lax"  
705 minOccurs="0" maxOccurs="unbounded"/>  
706 </sequence>  
707 </complexType>
```

708 3. Protocol

709 SAML assertions MAY be generated and exchanged using a variety of protocols. The bindings
710 and profiles specification for SAML [SAMLBind] describes specific means of transporting
711 assertions using existing widely deployed protocols.

712 SAML-aware clients MAY in addition use the SAML request-response protocol defined by the
713 <Request> and <Response> elements. The client sends a <Request> element to a SAML
714 service, and the service generates a <Response> element, as shown in Figure 1.



715

716

Figure 1: SAML Request-Response Protocol

717 3.1. Schema Header and Namespace Declarations

718 The following schema fragment defines the XML namespaces and other header information for
719 the protocol schema.

```
720 <schema  
721   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-  
722   sstc-schema-protocol-21.xsd"  
723   xmlns="http://www.w3.org/2001/XMLSchema"  
724   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
725   schema-protocol-21.xsd"  
726   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
727   schema-assertion-21.xsd"  
728   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
729   elementFormDefault="unqualified">  
730   <import namespace="http://www.oasis-open.org/committees/security/docs/draft-  
731   sstc-schema-assertion-21.xsd"  
732     schemaLocation="draft-sstc-schema-assertion-21.xsd"/>  
733   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
734     schemaLocation="xmldsig-core-schema.xsd"/>  
735   <annotation>  
736     <documentation>draft-sstc-schema-protocol-21.xsd</documentation>  
737   </annotation>  
738   ...  
739 </schema>
```

740 3.2. Simple Types

741 The following sections define the SAML protocol-related simple types.

742 3.2.1. Simple Type CompletenessSpecifierType

743 The **CompletenessSpecifierType** simple type is used in an attribute query request to filter the
744 response in cases where the requester is not authorized to receive part or all of the response.
745 The type enumerates the following possible values:

746 Partial

747 If there are any parts of the response that the requester is not authorized to receive, the requester
748 is asking for a response containing just those parts of the response that the requester is
749 authorized to receive.

750 AllOrNone
751 If there are any parts of the response that the requester is not authorized to receive, the requester
752 is asking for a response containing no assertions.

753 The following schema fragment defines the **CompletenessSpecifierType** simple type:

```
754 <simpleType name="CompletenessSpecifierType">  
755   <restriction base="string">  
756     <enumeration value="Partial"/>  
757     <enumeration value="AllOrNone"/>  
758   </restriction>  
759 </simpleType>
```

760 3.2.2. Simple Type StatusCodeType

761 The **StatusCodeType** simple type is used in a response to specify the status of the request that
762 caused the response to be generated. The type enumerates the following possible values:

763 Success

764 The request succeeded.

765 Failure

766 The request could not be performed by the service.

767 Error

768 An error in the request prevented the service from processing it.

769 Unknown

770 The request failed for unknown reasons[PHB6]

771 The following schema fragment defines the **StatusCodeType** simple type:

```
772 <simpleType name="StatusCodeType">  
773   <restriction base="string">  
774     <enumeration value="Success"/>  
775     <enumeration value="Failure"/>  
776     <enumeration value="Error"/>  
777     <enumeration value="Unknown"/>  
778   </restriction>  
779 </simpleType>
```

780 3.3. Requests

781 The following sections define the SAML constructs that contain request information.

782 3.3.1. Complex Type RequestAbstractType

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

785 RequestID [Required]

786 An identifier for the request. The values of the RequestID attribute in a request and the
787 InResponseTo attribute in the corresponding response MUST match.

788 MajorVersion [Required]

789 Each request MUST specify the SAML major version identifier. The identifier for this version of
790 SAML is 1. Processing of this attribute is specified in Section 4.

791 MinorVersion [Required]

792 Each request MUST specify the SAML minor version identifier. The identifier for this version of
793 SAML is 0. Processing of this attribute is specified in Section 4.

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

```

795 <complexType name="RequestAbstractType" abstract="true">
796   <attribute name="RequestID" type="saml:IDType" use="required"/>
797   <attribute name="MajorVersion" type="integer" use="required"/>
798   <attribute name="MinorVersion" type="integer" use="required"/>
799 </complexType>

```

800 3.3.2. Element <Request>

801 The <Request> element specifies a SAML request. It provides either a query or a request for a
802 specific assertion identified by <AssertionID> or <AssertionArtifact>. It has the complex
803 type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the
804 following elements:

805 <Query>

806 An extension point that allows extension schemas to define new types of query.

807 <SubjectQuery>

808 An extension point that allows extension schemas to define new types of query that specify a
809 single SAML subject.

810 <AuthenticationQuery>

811 Makes a query for authentication information.

812 <AttributeQuery>

813 Makes a query for attribute information.

814 <AuthorizationQuery>

815 Makes a query for an authorization decision.

816 <AssertionID> [One or more]

817 Requests an assertion by reference to its assertion identifier.

818 <AssertionArtifact> [One or more]

819 Requests an assertion by supplying an assertion artifact that represents it.

820 The following schema fragment defines the <Request> element and its **RequestType** complex
821 type:

```

822 <element name="Request" type="samlp:RequestType"/>
823 <complexType name="RequestType">
824   <complexContent>
825     <extension base="samlp:RequestAbstractType">
826       <choice>
827         <element ref="samlp:Query"/>
828         <element ref="samlp:SubjectQuery"/>
829         <element ref="samlp:AuthenticationQuery"/>
830         <element ref="samlp:AttributeQuery"/>
831         <element ref="samlp:AuthorizationQuery"/>
832         <element ref="saml:AssertionID" maxOccurs="unbounded"/>
833         <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
834       </choice>
835     </extension>
836   </complexContent>
837 </complexType>
838 <element name="AssertionArtifact" type="string"/>

```

839 3.4. Queries

840 The following sections define the SAML constructs that contain query information.

841 3.4.1. Element <Query>

842 The <Query> element is an extension point that allows new SAML queries to be defined. Its
843 **QueryAbstractType** is abstract; extension elements MUST use the `xsi:type` attribute to
844 indicate the derived type. **QueryAbstractType** is the base type from which all SAML query
845 elements are derived.

846 The following schema fragment defines the <Query> element and its **QueryAbstractType**
847 complex type:

```
848 <element name="Query" type="samlp:QueryAbstractType"/>  
849 <complexType name="QueryAbstractType" abstract="true"/>
```

850 3.4.2. Element <SubjectQuery>

851 The <SubjectQuery> element is an extension point that allows new SAML queries that specify
852 a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension
853 elements MUST use the `xsi:type` attribute to indicate the derived type.

854 **SubjectQueryAbstractType** adds the <Subject> element.

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

```
857 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
858 <complexType name="SubjectQueryAbstractType" abstract="true">  
859 <complexContent>  
860 <extension base="samlp:QueryAbstractType">  
861 <sequence>  
862 <element ref="saml:Subject"/>  
863 </sequence>  
864 </extension>  
865 </complexContent>  
866 </complexType>
```

867 3.4.3. Element <AuthenticationQuery>

868 The <AuthenticationQuery> element is used to make the query “What authentication
869 assertions are available for this subject?” The response will be in the form of an assertion
870 containing an authentication statement. This element is of type **AuthenticationQueryType**,
871 which extends **SubjectQueryAbstractType** with the addition of the following element:

872 <ConfirmationMethod> [Optional]

873 A filter for possible responses. If it is present, the query made is “What authentication assertions
874 do you have for this subject with the supplied confirmation method?”

875 In response to an authentication query, a responder returns assertions with authentication
876 statements as follows: The <Subject> element in the returned assertions MUST be identical to
877 the <Subject> element of the query. If the <ConfirmationMethod> element is present in the
878 query, at least one <ConfirmationMethod> element in the response MUST match. It is
879 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

880 The following schema fragment defines the <AuthenticationQuery> type and its
881 **AuthenticationQueryType** complex type:

```
882 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>  
883 <complexType name="AuthenticationQueryType">  
884 <complexContent>  
885 <extension base="samlp:SubjectQueryAbstractType">  
886 <sequence>  
887 <element ref="saml:ConfirmationMethod" minOccurs="0"/>  
888 </sequence>  
889 </extension>
```

```
890     </complexContent>
891 </complexType>
```

892 3.4.4. Element <AttributeQuery>

893 The <AttributeQuery> element is used to make the query “Return the requested attributes for
894 this subject.” The response will be in the form of an assertion containing an attribute statement.
895 This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with
896 the addition of the following element and attribute:

897 **CompletenessSpecifier** [Required] (see Section 0)

898 Filters the response in cases where the requester is not authorized to receive part or all of the
899 response. If its value is `Partial`, the query made is “Return any requested attributes for this
900 subject that this requester is authorized to see.” If its value is `AllOrNone`, the query made is
901 “Return the requested attributes for this subject, if and only if this requester is authorized to see
902 all of them.”

903 <AttributeDesignator> [Zero or more] (see Section 2.4.5.1)

904 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If
905 no attributes are specified, the list of desired attributes is implicit and application-specific.

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

```
908     <element name="AttributeQuery" type="saml:AttributeQueryType"/>
909     <complexType name="AttributeQueryType">
910       <complexContent>
911         <extension base="saml:SubjectQueryAbstractType">
912           <sequence>
913             <element ref="saml:AttributeDesignator"
914               minOccurs="0" maxOccurs="unbounded"/>
915           </sequence>
916           <attribute name="CompletenessSpecifier"
917             type="saml:CompletenessSpecifierType" use="required"/>
918         </extension>
919       </complexContent>
920     </complexType>
```

921 3.4.5. Element <AuthorizationQuery>

922 The <AuthorizationQuery> element is used to make the query “Should these actions on this
923 resource be allowed for this subject, given this evidence?” The response will be in the form of an
924 assertion containing an authorization decision statement. This element is of type
925 **AuthorizationQueryType**, which extends **SubjectQueryAbstractType** with the addition of the
926 following elements and attribute:

927 **Resource** [Required]

928 A URI indicating the resource for which authorization is requested.

929 <Actions> [Required]

930 The actions for which authorization is requested.

931 <Evidence> [Zero or more]

932 An assertion that the responder MAY rely on in making its response.

933 The following schema fragment defines the <AuthorizationQuery> element and its
934 **AuthorizationQueryType** complex type:

```
935     <element name="AuthorizationQuery" type="saml:AuthorizationQueryType"/>
936     <complexType name="AuthorizationQueryType">
937       <complexContent>
938         <extension base="saml:SubjectQueryAbstractType">
```

```

939     <sequence>
940         <element ref="saml:Actions"/>
941         <element ref="saml:Evidence"
942             minOccurs="0" maxOccurs="unbounded"/>
943     </sequence>
944     <attribute name="Resource" type="anyURI"/>
945 </extension>
946 </complexContent>
947 </complexType>

```

948 3.5. Responses

949 The following sections define the SAML constructs that contain response information.

950 3.5.1. Complex Type ResponseAbstractType

951 All SAML responses are of types that are derived from the abstract **ResponseAbstractType**
952 complex type. This type defines common attributes that are associated with all SAML responses:

953 **ResponseID** [Required]

954 An identifier for the response.

955 **InResponseTo** [Required]

956 A reference to the identifier of the request to which the response corresponds. The value of this
957 attribute **MUST** match the value of the corresponding `RequestID` attribute.

958 **MajorVersion** [Required]

959 Each response **MUST** specify the SAML major version identifier. The identifier for this version of
960 SAML is 1. Processing of this attribute is specified in Section 4.

961 **MinorVersion** [Required]

962 Each response **MUST** specify the SAML minor version identifier. The identifier for this version of
963 SAML is 0. Processing of this attribute is specified in Section 4.

964 The following schema fragment defines the **ResponseAbstractType** complex type:

```

965 <complexType name="ResponseAbstractType" abstract="true">
966     <attribute name="ResponseID" type="saml:IDType" use="required"/>
967     <attribute name="InResponseTo" type="saml:IDType" use="required"/>
968     <attribute name="MajorVersion" type="integer" use="required"/>
969     <attribute name="MinorVersion" type="integer" use="required"/>
970 </complexType>

```

971 3.5.2. Element <Response>

972 The `<Response>` element specifies the status of the corresponding SAML request and a list of
973 zero or more assertions that answer the request. It has the complex type **ResponseType**, which
974 extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture)
975 and attribute:

976 **StatusCode** [Required] (see Section 3.2.2)

977 A code representing the status of the corresponding request.

978 `<Assertion>` (see Section 2.3.3)

979 Specifies an assertion by value.

980 `<SingleAssertion>`

981 Specifies an assertion containing a single statement by value.

982 `<MultipleAssertion>`

983 Specifies an assertion containing multiple statements by value.

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

```
986 <element name="Response" type="samlp:ResponseType"/>
987 <complexType name="ResponseType">
988   <complexContent>
989     <extension base="samlp:ResponseAbstractType">
990       <choice minOccurs="0" maxOccurs="unbounded">
991         <element ref="saml:Assertion"/>
992         <element ref="saml:SingleAssertion"/>
993         <element ref="saml:MultipleAssertion"/>
994       </choice>
995       <attribute name="StatusCode"
996                 type="samlp:StatusCodeType" use="required"/>
997     </extension>
998   </complexContent>
999 </complexType>
```

4. Assertion and Protocol Versioning

1000

1001 SAML version information appears in the following elements:

- 1002 • <Assertion>
- 1003 • <Request>
- 1004 • <Response>

1005 The version numbering of the SAML assertion is independent of the version number of the SAML
1006 request-response protocol. The version information for each consists of a major version number
1007 and a minor version number, both of which are integers. In accordance with industry practice a
1008 version number SHOULD be presented to the user in the form *Major.Minor*. This document
1009 defines SAML Assertions 1.0 and SAML Protocol 1.0.

1010 The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

1011 $Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B = Minor_A)$

1012 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses
1013 that are the same as or higher than the corresponding version number in the SAML version that
1014 immediately preceded it.

1015 New versions of SAML SHALL assign new version numbers as follows:

- 1016 • **Documentation change:** $(Major_B = Major_A) \wedge (Minor_B = Minor_A)$
1017 If the major and minor version numbers are unchanged, the new version *B* only
1018 introduces changes to the documentation that raise no compatibility issues with an
1019 implementation of version *A*.
- 1020 • **Minor upgrade:** $(Major_B = Major_A) \wedge (Minor_B > Minor_A)$
1021 If the major version number of versions *A* and *B* are the same and the minor version
1022 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to
1023 the SAML schema and semantics but any changes that are introduced in *B* SHALL be
1024 compatible with version *A*.
- 1025 • **Major upgrade:** $Major_B > Major_A$
1026 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY
1027 introduce changes to the SAML schema and semantics that are incompatible with *A*.

4.1. Assertion Version

1028 A SAML application MUST NOT issue any assertion whose version number is not supported.

1029 A SAML application MUST reject any assertion whose major version number is not supported.

1030 A SAML application MAY reject any assertion whose version number is higher than the highest
1031 supported version.
1032

4.2. Request Version

1033 A SAML application SHOULD issue requests that specify the highest SAML version supported by
1034 both the sender and recipient.
1035

1036 If the SAML application does not know the capabilities of the recipient it should assume that it
1037 supports the highest SAML version supported by the sender.

1038 **4.3. Response Version**

1039 A SAML application MUST NOT issue responses that specify a higher SAML version number
1040 than the corresponding request.

1041 A SAML application MUST NOT issue a response that has a major version number that is lower
1042 than the major version number of the corresponding request except to report the error

1043 `RequestVersionTooHigh`.

1044 Incompatible protocol versions MAY cause the following errors to be reported:

1045 `RequestVersionTooHigh`

1046 The protocol version specified in the request is a major upgrade from the highest protocol version
1047 supported by the responder.

1048 `RequestVersionTooLow`

1049 The responder cannot respond to the particular request using the SAML version specified in the
1050 request because it is too low.

1051 `RequestVersionDeprecated`

1052 The responder does not respond to any requests with the protocol version specified in the
1053 request.

1054 **5. Schema Extension**

1055 The SAML schemas support extensibility. An example of an application that extends SAML
1056 assertions is the XTAML system for management of embedded trust roots [XTAML]. The
1057 following sections explain how to use the extensibility features in SAML to create extension
1058 schemas.

1059 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML
1060 elements MAY serve as the head element of a substitution group. Also, types are not defined as
1061 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss
1062 only elements that have been specifically designed to support extensibility.

1063 **5.1. Assertion Schema Extension**

1064 The SAML assertion schema is designed to permit separate processing of the assertion package
1065 and the statements it contains, if the extension mechanism is used for either part.

1066 The following elements are intended specifically for use as extension points in an extension
1067 schema; their types are set to *abstract*, so that the use of an `xsi:type` attribute with these
1068 elements is REQUIRED:

- 1069 • `<Assertion>`
- 1070 • `<Statement>`
- 1071 • `<SubjectStatement>`
- 1072 • `<AttributeValue>`
- 1073 • `<Condition>`
- 1074 • `<AdviceElement>`

1075 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1076 • `<SingleAssertion>`
- 1077 • `<MultipleAssertion>`
- 1078 • `<AuthenticationStatement>`
- 1079 • `<AuthorizationStatement>`
- 1080 • `<AttributeStatement>`
- 1081 • `<AudienceRestrictionCondition>`

1082 **5.2. Protocol Schema Extension**

1083 The following elements are intended specifically for use as extension points in an extension
1084 schema; their types are set to *abstract*, so that the use of an `xsi:type` attribute with these
1085 elements is REQUIRED:

- 1086 • `<Request>`
- 1087 • `<Query>`
- 1088 • `<SubjectQuery>`
- 1089 • `<Response>`

1090 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1091 • <AuthenticationQuery>
- 1092 • <AuthorizationQuery>
- 1093 • <AttributeQuery>

1094 **5.3. Use of Type Derivation and Substitution Groups**

1095 W3C XML Schema [**Schema1**] provides two principal mechanisms for specifying an element of
1096 an extended type: type derivation and substitution groups.

1097 For example, a <Statement> element can be assigned the type **NewStatementType** by means
1098 of the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType**
1099 needs to be derived from **StatementType**. The following example of a SAML assertion assumes that the
1100 extension schema (represented by the `new:` prefix) has defined this new type:

```
1101 <saml:Assertion ...>  
1102   <saml:Statement xsi:type="new:NewStatementType">  
1103     ...  
1104   </saml:Statement>  
1105 </saml:Assertion>
```

1106 Alternatively, the extension schema can define a <NewStatement> element that is a member of
1107 a substitution group that has <Statement> as a head element. For the substituted element to be
1108 schema-valid, it needs to have a type that matches or is derived from the head element's type.
1109 The following is an example of an extension schema fragment that defines this new element:

```
1110 <xsd:element "NewStatement" type="new:NewStatementType"  
1111   substitutionGroup="saml:Statement"/>
```

1112 The substitution group declaration allows the <NewStatement> element to be used anywhere
1113 the SAML <Statement> element can be used. The following is an example of a SAML assertion
1114 that uses the extension element:

```
1115 <saml:Assertion ...>  
1116   <new:NewStatement>  
1117     ...  
1118   </new:NewStatement>  
1119 </saml:Assertion>
```

1120 The choice of extension method has no effect on the semantics of the XML document but does
1121 have implications for interoperability.

1122 The advantages of type derivation are as follows:

- 1123 • A document can be more fully interpreted by a parser that does not have access to the
1124 extension schema because a "native" SAML element is available.
- 1125 • At the time of writing, some W3C XML Schema validators do not support substitution
1126 groups, whereas the `xsi:type` attribute is widely supported.

1127 The advantage of substitution groups is that a document can be explained without the need to
1128 explain the functioning of the `xsi:type` attribute.

1129 **6. Identifiers**

1130 The following sections define URI-based identifiers for common authentication protocols and
1131 actions.

1132 [TBD][elm7]

1133 **6.1. Confirmation Method Identifiers**

1134 The following identifiers MAY be used in the `<ConfirmationMethod>` element (see Section
1135 2.4.2.3.1) to refer to common authentication protocols.

1136 SAML Artifact:
1137 [TBD]

1138 Assertion Bearer:
1139 [TBD]

1140 User Name and Password (Pass-Through):
1141 [TBD]

1142 User Name and Password (One-Way-Function SHA-1):
1143 [TBD]

1144 Kerberos [**Kerberos**]:
1145 [TBD]

1146 SSL/TLS Certificate Based Client Authentication:
1147 [TBD]

1148 Object Authenticator (SHA-1):
1149 [TBD]

1150 **6.2. Action Namespace Identifiers**

1151 The following identifiers MAY be used in the `ActionNamespace` attribute (see Section 2.4.4.1)
1152 to refer to common sets of actions to perform on resources.

1153 Read/Write/Execute/Delete/Control:
1154 [TBD]

1155 Read/Write/Execute/Delete/Control with Negation:
1156 [TBD]

1157 Get/Head/Put/Post:
1158 [TBD]

1159 UNIX File Permissions:
1160 [TBD]

1161

7. Schema Listings

1162

The following sections contain complete listings of the assertion and protocol schemas for SAML.

1163

[TBD][elm8]

1164

7.1. Assertion Schema

1165

Following is a complete listing of the SAML assertion schema [SAML-XSD].

1166

```
<?xml version="1.0" encoding="UTF-8"?>
```

1167

```
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
```

1168

```
(VeriSign Inc.) -->
```

1169

```
<schema
```

1170

```
  targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
```

1171

```
sstc-schema-assertion-19.xsd"
```

1172

```
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#" 
```

1173

```
  xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
```

1174

```
schema-assertion-21.xsd"
```

1175

```
  xmlns="http://www.w3.org/2001/XMLSchema" 
```

1176

```
  elementFormDefault="unqualified">
```

1177

```
  <import namespace="http://www.w3.org/2000/09/xmldsig#" 
```

1178

```
    schemaLocation="xmldsig-core-schema.xsd"/>
```

1179

```
  <annotation>
```

1180

```
    <documentation>draft-sstc-schema-assertion-19.xsd</documentation>
```

1181

```
  </annotation>
```

1182

```
  <element name="AssertionID" type="saml:IDType"/>
```

1183

```
  <simpleType name="IDType">
```

1184

```
    <restriction base="string"/>
```

1185

```
  </simpleType>
```

1186

```
  <simpleType name="DecisionType">
```

1187

```
    <restriction base="string">
```

1188

```
      <enumeration value="Permit"/>
```

1189

```
      <enumeration value="Deny"/>
```

1190

```
      <enumeration value="Indeterminate"/>
```

1191

```
    </restriction>
```

1192

```
  </simpleType>
```

1193

```
  <element name="Assertion" type="saml:AssertionAbstractType"/>
```

1194

```
  <complexType name="AssertionAbstractType" abstract="true">
```

1195

```
    <sequence>
```

1196

```
      <element ref="saml:Conditions" minOccurs="0"/>
```

1197

```
      <element ref="saml:Advice" minOccurs="0"/>
```

1198

```
    </sequence>
```

1199

```
    <attribute name="MajorVersion" type="integer" use="required"/>
```

1200

```
    <attribute name="MinorVersion" type="integer" use="required"/>
```

1201

```
    <attribute name="AssertionID" type="saml:IDType" use="required"/>
```

1202

```
    <attribute name="Issuer" type="string" use="required"/>
```

1203

```
    <attribute name="IssueInstant" type="dateTime" use="required"/>
```

1204

```
  </complexType>
```

1205

```
  <element name="SingleAssertion" type="saml:SingleAssertionType"/>
```

1206

```
  <complexType name="SingleAssertionType">
```

1207

```
    <complexContent>
```

1208

```
      <extension base="saml:AssertionAbstractType">
```

1209

```
        <choice>
```

1210

```
          <element ref="saml:Statement"/>
```

1211

```
          <element ref="saml:SubjectStatement"/>
```

1212

```
          <element ref="saml:AuthenticationStatement"/>
```

1213

```
          <element ref="saml:AuthorizationStatement"/>
```

1214

```
          <element ref="saml:AttributeStatement"/>
```

1215

```
        </choice>
```

1216

```
      </extension>
```

1217

```
    </complexContent>
```

```

1218 </complexType>
1219 <element name="MultipleAssertion" type="saml:MultipleAssertionType"/>
1220 <complexType name="MultipleAssertionType">
1221   <complexContent>
1222     <extension base="saml:AssertionAbstractType">
1223       <choice minOccurs="0" maxOccurs="unbounded">
1224         <element ref="saml:Statement"/>
1225         <element ref="saml:SubjectStatement"/>
1226         <element ref="saml:AuthenticationStatement"/>
1227         <element ref="saml:AuthorizationStatement"/>
1228         <element ref="saml:AttributeStatement"/>
1229       </choice>
1230     </extension>
1231   </complexContent>
1232 </complexType>
1233 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
1234 <complexType name="AssertionSpecifierType">
1235   <choice>
1236     <element ref="saml:AssertionID"/>
1237     <element ref="saml:Assertion"/>
1238     <element ref="saml:SingleAssertion"/>
1239     <element ref="saml:MultipleAssertion"/>
1240   </choice>
1241 </complexType>
1242 <element name="Statement" type="saml:StatementAbstractType"/>
1243 <complexType name="StatementAbstractType" abstract="true"/>
1244 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1245 <complexType name="SubjectStatementAbstractType" abstract="true">
1246   <complexContent>
1247     <extension base="saml:StatementAbstractType">
1248       <sequence>
1249         <element ref="saml:Subject"/>
1250       </sequence>
1251     </extension>
1252   </complexContent>
1253 </complexType>
1254 <element name="Subject" type="saml:SubjectType"/>
1255 <complexType name="SubjectType">
1256   <choice maxOccurs="unbounded">
1257     <element ref="saml:NameIdentifier"/>
1258     <element ref="saml:SubjectConfirmation"/>
1259     <element ref="saml:AssertionSpecifier"/>
1260   </choice>
1261 </complexType>
1262 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1263 <complexType name="SubjectConfirmationType">
1264   <sequence>
1265     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1266     <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
1267     <element ref="ds:KeyInfo" minOccurs="0"/>
1268   </sequence>
1269   <!-- Need to modify this element-->
1270 </complexType>
1271 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1272 <complexType name="NameIdentifierType">
1273   <attribute name="SecurityDomain" type="string"/>
1274   <attribute name="Name" type="string"/>
1275 </complexType>
1276 <element name="ConfirmationMethod" type="anyURI"/>
1277 <element name="AuthenticationStatement"
1278   type="saml:AuthenticationStatementType"/>
1279 <complexType name="AuthenticationStatementType">
1280   <complexContent>

```



```

1281     <extension base="saml:SubjectStatementAbstractType">
1282         <sequence>
1283             <element ref="saml:AuthenticationLocality" minOccurs="0"/>
1284         </sequence>
1285         <attribute name="AuthenticationMethod" type="anyURI"/>
1286         <attribute name="AuthenticationInstant" type="dateTime"/>
1287     </extension>
1288 </complexContent>
1289 </complexType>
1290 <element name="AuthenticationLocality"
1291     type="saml:AuthenticationLocalityType"/>
1292 <complexType name="AuthenticationLocalityType">
1293     <attribute name="IPAddress" type="string" use="optional"/>
1294     <attribute name="DNSAddress" type="string" use="optional"/>
1295 </complexType>
1296 <element name="AuthorizationStatement"
1297     type="saml:AuthorizationStatementType"/>
1298 <complexType name="AuthorizationStatementType">
1299     <complexContent>
1300         <extension base="saml:SubjectStatementAbstractType">
1301             <sequence>
1302                 <element ref="saml:Actions"/>
1303                 <element ref="saml:Evidence"
1304                     minOccurs="0" maxOccurs="unbounded"/>
1305             </sequence>
1306             <attribute name="Resource" type="anyURI" use="optional"/>
1307             <attribute name="Decision"
1308                 type="saml:DecisionType" use="optional"/>
1309         </extension>
1310     </complexContent>
1311 </complexType>
1312 <element name="Actions" type="saml:ActionsType"/>
1313 <complexType name="ActionsType">
1314     <sequence>
1315         <element ref="saml:Action" maxOccurs="unbounded"/>
1316     </sequence>
1317     <attribute name="Namespace" type="anyURI" use="optional"/>
1318 </complexType>
1319 <element name="Action" type="string"/>
1320 <element name="Evidence" type="saml:AssertionSpecifierType"/>
1321 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1322 <complexType name="AttributeStatementType">
1323     <complexContent>
1324         <extension base="saml:SubjectStatementAbstractType">
1325             <sequence>
1326                 <element ref="saml:Attribute" maxOccurs="unbounded"/>
1327             </sequence>
1328         </extension>
1329     </complexContent>
1330 </complexType>
1331 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
1332 <complexType name="AttributeDesignatorType">
1333     <attribute name="AttributeName" type="string"/>
1334     <attribute name="AttributeNamespace" type="anyURI"/>
1335 </complexType>
1336 <element name="Attribute" type="saml:AttributeType"/>
1337 <complexType name="AttributeType">
1338     <complexContent>
1339         <extension base="saml:AttributeDesignatorType">
1340             <sequence>
1341                 <element ref="saml:AttributeValue"/>
1342             </sequence>
1343         </extension>

```

```

1344     </complexContent>
1345 </complexType>
1346 <element name="AttributeValue" type="saml:AttributeValueType"/>
1347 <complexType name="AttributeValueType">
1348   <sequence>
1349     <any namespace="##any" processContents="lax"
1350       minOccurs="0" maxOccurs="unbounded"/>
1351   </sequence>
1352 </complexType>
1353 <element name="Conditions" type="saml:ConditionsType"/>
1354 <complexType name="ConditionsType">
1355   <choice minOccurs="0" maxOccurs="unbounded">
1356     <element ref="saml:Condition"/>
1357     <element ref="saml:AudienceRestrictionCondition"/>
1358   </choice>
1359   <attribute name="NotBefore" type="dateTime" use="optional"/>
1360   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
1361 </complexType>
1362 <element name="Condition" type="saml:ConditionAbstractType"/>
1363 <complexType name="ConditionAbstractType" abstract="true"/>
1364 <element name="AudienceRestrictionCondition"
1365   type="saml:AudienceRestrictionConditionType"/>
1366 <complexType name="AudienceRestrictionConditionType">
1367   <complexContent>
1368     <extension base="saml:ConditionAbstractType">
1369       <sequence>
1370         <element ref="saml:Audience"
1371           minOccurs="1" maxOccurs="unbounded"/>
1372       </sequence>
1373     </extension>
1374   </complexContent>
1375 </complexType>
1376 <element name="Audience" type="anyURI"/>
1377 <element name="Advice" type="saml:AdviceType"/>
1378 <complexType name="AdviceType">
1379   <sequence>
1380     <choice minOccurs="0" maxOccurs="unbounded">
1381       <element ref="saml:AssertionSpecifier"/>
1382       <element ref="saml:AdviceElement"/>
1383       <any namespace="##other" processContents="lax"/>
1384     </choice>
1385   </sequence>
1386 </complexType>
1387 <element name="AdviceElement" type="saml:AdviceAbstractType"/>
1388 <complexType name="AdviceAbstractType"/>
1389 </schema>

```

7.2. Protocol Schema

1390

Following is a complete listing of the SAML protocol schema [SAML-XMLS].

1391

```

1392 <?xml version="1.0" encoding="UTF-8"?>
1393 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1394 (VeriSign Inc.) -->
1395 <schema
1396   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
1397   sstc-schema-protocol-21.xsd"
1398   xmlns="http://www.w3.org/2001/XMLSchema"
1399   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1400   schema-protocol-21.xsd"
1401   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1402   schema-assertion-21.xsd"
1403   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"

```

```

1404     elementFormDefault="unqualified">
1405     <import namespace="http://www.oasis-open.org/committees/security/docs/draft-
1406     sstc-schema-assertion-21.xsd"
1407     schemaLocation="draft-sstc-schema-assertion-21.xsd"/>
1408     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1409     schemaLocation="xmldsig-core-schema.xsd"/>
1410     <annotation>
1411         <documentation>draft-sstc-schema-protocol-21.xsd</documentation>
1412     </annotation>
1413     <simpleType name="CompletenessSpecifierType">
1414         <restriction base="string">
1415             <enumeration value="Partial"/>
1416             <enumeration value="AllOrNone"/>
1417         </restriction>
1418     </simpleType>
1419     <simpleType name="StatusCodeType">
1420         <restriction base="string">
1421             <enumeration value="Success"/>
1422             <enumeration value="Failure"/>
1423             <enumeration value="Error"/>
1424             <enumeration value="Unknown"/>
1425         </restriction>
1426     </simpleType>
1427     <complexType name="RequestAbstractType" abstract="true">
1428         <attribute name="RequestID" type="saml:IDType" use="required"/>
1429         <attribute name="MajorVersion" type="integer" use="required"/>
1430         <attribute name="MinorVersion" type="integer" use="required"/>
1431     </complexType>
1432     <element name="Request" type="samlp:RequestType"/>
1433     <complexType name="RequestType">
1434         <complexContent>
1435             <extension base="samlp:RequestAbstractType">
1436                 <choice>
1437                     <element ref="samlp:Query"/>
1438                     <element ref="samlp:SubjectQuery"/>
1439                     <element ref="samlp:AuthenticationQuery"/>
1440                     <element ref="samlp:AttributeQuery"/>
1441                     <element ref="samlp:AuthorizationQuery"/>
1442                     <element ref="saml:AssertionID" maxOccurs="unbounded"/>
1443                     <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
1444                 </choice>
1445             </extension>
1446         </complexContent>
1447     </complexType>
1448     <element name="AssertionArtifact" type="string"/>
1449     <element name="Query" type="samlp:QueryAbstractType"/>
1450     <complexType name="QueryAbstractType" abstract="true"/>
1451     <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
1452     <complexType name="SubjectQueryAbstractType" abstract="true">
1453         <complexContent>
1454             <extension base="samlp:QueryAbstractType">
1455                 <sequence>
1456                     <element ref="saml:Subject"/>
1457                 </sequence>
1458             </extension>
1459         </complexContent>
1460     </complexType>
1461     <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
1462     <complexType name="AuthenticationQueryType">
1463         <complexContent>
1464             <extension base="samlp:SubjectQueryAbstractType">
1465                 <sequence>
1466                     <element ref="saml:ConfirmationMethod" minOccurs="0"/>

```

```

1467         </sequence>
1468     </extension>
1469 </complexContent>
1470 </complexType>
1471 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1472 <complexType name="AttributeQueryType">
1473     <complexContent>
1474         <extension base="samlp:SubjectQueryAbstractType">
1475             <sequence>
1476                 <element ref="saml:AttributeDesignator"
1477                     minOccurs="0" maxOccurs="unbounded"/>
1478             </sequence>
1479             <attribute name="CompletenessSpecifier"
1480                 type="samlp:CompletenessSpecifierType" use="required"/>
1481         </extension>
1482     </complexContent>
1483 </complexType>
1484 <element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>
1485 <complexType name="AuthorizationQueryType">
1486     <complexContent>
1487         <extension base="samlp:SubjectQueryAbstractType">
1488             <sequence>
1489                 <element ref="saml:Actions"/>
1490                 <element ref="saml:Evidence"
1491                     minOccurs="0" maxOccurs="unbounded"/>
1492             </sequence>
1493             <attribute name="Resource" type="anyURI"/>
1494         </extension>
1495     </complexContent>
1496 </complexType>
1497 <complexType name="ResponseAbstractType" abstract="true">
1498     <attribute name="ResponseID" type="saml:IDType" use="required"/>
1499     <attribute name="InResponseTo" type="saml:IDType" use="required"/>
1500     <attribute name="MajorVersion" type="integer" use="required"/>
1501     <attribute name="MinorVersion" type="integer" use="required"/>
1502 </complexType>
1503 <element name="Response" type="samlp:ResponseType"/>
1504 <complexType name="ResponseType">
1505     <complexContent>
1506         <extension base="samlp:ResponseAbstractType">
1507             <choice minOccurs="0" maxOccurs="unbounded">
1508                 <element ref="saml:Assertion"/>
1509                 <element ref="saml:SingleAssertion"/>
1510                 <element ref="saml:MultipleAssertion"/>
1511             </choice>
1512             <attribute name="StatusCode"
1513                 type="samlp:StatusCodeType" use="required"/>
1514         </extension>
1515     </complexContent>
1516 </complexType>
1517 </schema>

```

1518 8. References

1519 [TBD]^[elm9]

- 1520 **[Kerberos]** R. Needham et al., *Using Encryption for Authentication in Large*
1521 *Networks of Computers*, Communications of the ACM, Vol. 21 (12), pp.
1522 993-999, December 1978.
- 1523 **[PKCS1]** B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories,
1524 also IETF RFC 2437, October 1998.
- 1525 **[RFC2104]** H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,
1526 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 1527 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1528 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 1529 **[SAML-XSD]** P. Hallam-Baker et al., *SAML assertion schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd)
1530 [open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd),
1531 OASIS, December 2001.
- 1532 **[SAMLBind]** P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion*
1533 *Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf)
1534 [open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf),
1535 OASIS, December 2001.
- 1536 **[SAMLGloss]** J. Hodges et al., *Glossary for the OASIS Security Assertion Markup*
1537 *Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)
1538 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,
1539 December 2001.
- 1540 **[SAML-P-XSD]** P. Hallam-Baker et al., *SAML protocol schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd)
1541 [open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd),
1542 OASIS, December 2001.
- 1543 **[Schema1]** H. S. Thompson et al., *XML Schema Part 1: Structures*,
1544 <http://www.w3.org/TR/xmlschema-1/>, World Wide Web Consortium
1545 Recommendation, May 2001.
- 1546 **[Schema2]** P. V. Biron et al., *XML Schema Part 2: Datatypes*,
1547 <http://www.w3.org/TR/xmlschema-2/>, World Wide Web Consortium
1548 Recommendation, May 2001.
- 1549 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*,
1550 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.
- 1551 **[XMLSig-XSD]** XML Signature Schema available from [http://www.w3.org/TR/2000/CR-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)
1552 [xmlsig-core-20001031/xmlsig-core-schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).
- 1553 **[XMLEnc]** *XML Encryption Specification*, In development.
- 1554 **[XTAML]** P. Hallam-Baker, *XML Trust Axiom Markup Language 1.0*,
1555 <http://www.xmltrustcenter.org/>, VeriSign Inc. September 2001.

1556

Appendix A. Notices

1557 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1558 that might be claimed to pertain to the implementation or use of the technology described in this
1559 document or the extent to which any license under such rights might or might not be available;
1560 neither does it represent that it has made any effort to identify any such rights. Information on
1561 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1562 website. Copies of claims of rights made available for publication and any assurances of licenses
1563 to be made available, or the result of an attempt made to obtain a general license or permission
1564 for the use of such proprietary rights by implementors or users of this specification, can be
1565 obtained from the OASIS Executive Director.

1566 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1567 applications, or other proprietary rights which may cover technology that may be required to
1568 implement this specification. Please address the information to the OASIS Executive Director.

1569 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
1570 2001. All Rights Reserved.

1571 This document and translations of it may be copied and furnished to others, and derivative works
1572 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1573 published and distributed, in whole or in part, without restriction of any kind, provided that the
1574 above copyright notice and this paragraph are included on all such copies and derivative works.
1575 However, this document itself may not be modified in any way, such as by removing the copyright
1576 notice or references to OASIS, except as needed for the purpose of developing OASIS
1577 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1578 Property Rights document must be followed, or as required to translate it into languages other
1579 than English.

1580 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1581 successors or assigns.

1582 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1583 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1584 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1585 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1586 PARTICULAR PURPOSE.

Page: 1

[elm1]Check this list. Should we just list editors? List whole TC? List whoever wants to be listed?

Page: 6

[elm2]Check: This is old!

Page: 6

[elm3]Need conceptual material here. Explain concepts/terms such as the domain model, URIs for identifiers, extension points, etc.

Page: 9

[PHB4] Need some better text here n'est pas?

Page: 17

[elm5] don't know what's really supposed to go here.

Page: 20

[PHB6]Need to have text for these, how exactly does failure differ from error?

Page: 30

[elm7]Need to get these filled in.

Page: 31

[elm8]The references to the XML Signature schema are outdated. In the SAML schemas, and throughout the spec, this needs to be fixed.

Page: 37

[elm9]: Need to check and sort all references.