



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

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

Document identifier: draft-sstc-core-23

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

Publication date: 4th January, 2002

Status: Interim draft. Send comments to the editors.

Editors:

- Phillip Hallam-Baker, VeriSign, (pbaker@verisign.com)
- Eve Maler, Sun Microsystems, (eve.maler@sun.com)
- Krishna Sankar, Cisco Systems Inc(ksankar@cisco.com)

Contributors:

- Carlisle Adams, Entrust
- Scott Cantor, osu.edu
- Marc Chanliau, Netegrity
- Nigel Edwards, Hewlett-Packard
- Marlena Erdos, Tivoli
- Stephen Farrell, Baltimore
- Simon Godik, Crosslogic
- Jeff Hodges, Oblix
- Charles Knouse, Oblix
- Chris McLaren, Netegrity
- Prateek Mishra, Netegrity
- RL "Bob" Morgan, University of Washington
- Tim Moses, Entrust
- David Orchard, BEA
- Joe Pato, Hewlett Packard
- Darren Platt, RSA
- Irving Reid, Baltimore

31		
32	ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP	
33	LANGUAGE (SAML)	1
34	1. INTRODUCTION	6
35	1.1. NOTATION	6
36	1.2. SCHEMA ORGANIZATION AND NAMESPACES	6
37	1.3. SAML CONCEPTS (NON-NORMATIVE)	7
38	2. SAML ASSERTIONS	8
39	2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	8
40	2.2. SIMPLE TYPES	8
41	2.2.1. <i>Simple Type IDType</i>	8
42	2.2.2. <i>Simple Type DecisionType</i>	9
43	2.3. ASSERTIONS	9
44	2.3.1. <i>Element <AssertionSpecifier></i>	9
45	2.3.2. <i>Element <AssertionID></i>	9
46	2.3.3. <i>Element <Assertion></i>	10
47	2.3.3.1. <i>Element <Conditions></i>	11
48	2.3.3.1.1. <i>Attributes NotBefore and NotOnOrAfter</i>	12
49	2.3.3.1.2. <i>Element <Condition></i>	12
50	2.3.3.1.3. <i>Elements <AudienceRestrictionCondition> and <Audience></i>	12
51	2.3.3.1.4. <i>Condition Type TargetRestrictionType</i>	13
52	2.3.3.2. <i>Elements <Advice> and <AdviceElement></i>	13
53	2.4. STATEMENTS	14
54	2.4.1. <i>Element <Statement></i>	14
55	2.4.2. <i>Element <SubjectStatement></i>	14
56	2.4.2.1. <i>Element <Subject></i>	14
57	2.4.2.2. <i>Element <NameIdentifier></i>	15
58	2.4.2.3. <i>Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData></i>	15
59	2.4.3. <i>Element <AuthenticationStatement></i>	16

60	2.4.3.1. Element <AuthenticationLocality>	16
61	2.4.4. Element <AuthorizationDecisionStatement>	17
62	2.4.4.1. Elements <Actions> and <Action>	17
63	2.4.4.2. Element <Evidence>	18
64	2.4.5. Element <AttributeStatement>	18
65	2.4.5.1. Elements <AttributeDesignator> and <Attribute>	18
66	2.4.5.1.1 Element <AttributeValue>	19
67	3. SAML PROTOCOL	20
68	3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	20
69	3.2. SIMPLE TYPES	20
70	3.2.1. Simple Type StatusCodeType	20
71	3.3. REQUESTS	21
72	3.3.1. Complex Type RequestAbstractType	21
73	3.3.1.1. Element <RespondWith>	21
74	3.3.2. Element <Request>	22
75	3.4. QUERIES	23
76	3.4.1. Element <Query>	23
77	3.4.2. Element <SubjectQuery>	23
78	3.4.3. Element <AuthenticationQuery>	23
79	3.4.4. Element <AttributeQuery>	24
80	3.4.5. Element <AuthorizationDecisionQuery>	24
81	3.5. RESPONSES	25
82	3.5.1. Complex Type ResponseAbstractType	25
83	3.5.2. Element <Response>	25
84	3.5.2.1. Element <StatusReason>	26
85	4. SAML VERSIONING	27
86	4.1. ASSERTION VERSION	27
87	4.2. REQUEST VERSION	27

88	4.3. RESPONSE VERSION	28
89	5. SAML & XML-SIGNATURE SYNTAX AND PROCESSING	29
90	5.1. SIGNING ASSERTIONS	30
91	5.2. REQUEST /RESPONSE SIGNING	30
92	5.3. SIGNATURE INHERITANCE (A.K.A. SUPER-SIGNATURES & SUB-MESSAGES)	30
93	5.3.1. <i>Rationale</i>	30
94	5.3.2. <i>Rules for SAML Signature Inheritance</i>	30
95	5.4. XML SIGNATURE PROFILE	31
96	5.4.1. <i>Signing formats</i>	31
97	5.4.2. <i>CanonicalizationMethod</i>	31
98	5.4.3. <i>Transforms</i>	31
99	5.4.4. <i>KeyInfo</i>	31
100	5.4.5. <i>Binding between statements in a multi-statement assertion</i>	31
101	5.4.6. <i>Security considerations</i>	31
102	5.4.6.1. <i>Replay Attack</i>	31
103	6. SAML EXTENSIONS	32
104	6.1. ASSERTION SCHEMA EXTENSION	32
105	6.2. PROTOCOL SCHEMA EXTENSION	32
106	6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	33
107	7. SAML-DEFINED IDENTIFIERS	34
108	7.1. CONFIRMATION METHOD IDENTIFIERS	34
109	7.1.1. <i>SAML Artifact:</i>	34
110	7.1.2. <i>SAML Artifact (SHA-1):</i>	34
111	7.1.3. <i>Holder of Key:</i>	34
112	7.1.4. <i>Sender Vouches:</i>	34

113	<i>7.1.5. Password (Pass-Through):</i>	34
114	<i>7.1.6. Password (One-Way-Function SHA-1):</i>	35
115	<i>7.1.7. Kerberos [Kerberos]</i>	35
116	<i>7.1.8. SSL/TLS Certificate Based Client Authentication:</i>	35
117	<i>7.1.9. Object Authenticator (SHA-1):</i>	35
118	<i>7.1.10. PKCS#7</i>	35
119	<i>7.1.11. Cryptographic Message Syntax</i>	36
120	<i>7.1.12. XML Digital Signature</i>	36
121	7.2. ACTION NAMESPACE IDENTIFIERS	36
122	<i>7.2.1. Read/Write/Execute/Delete/Control:</i>	36
123	<i>7.2.2. Read/Write/Execute/Delete/Control with Negation:</i>	36
124	<i>7.2.3. Get/Head/Put/Post:</i>	37
125	<i>7.2.4. UNIX File Permissions:</i>	37
126	8. SAML SCHEMA LISTINGS	38
127	8.1. ASSERTION SCHEMA	38
128	8.2. PROTOCOL SCHEMA	41
129	9. REFERENCES	44
130	APPENDIX A. NOTICES	46
131		

132 1. Introduction

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

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

140 1.1. Notation

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

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

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

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

153 Listings of SAML schemas appear like this.

154 `Example code listings appear like this.`

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

- 159 • The prefix `saml`: stands for the SAML assertion namespace.
- 160 • The prefix `samlp`: stands for the SAML request-response protocol namespace.
- 161 • The prefix `ds`: stands for the W3C XML Signature namespace.
- 162 • The prefix `xsd`: stands for the W3C XML Schema namespace in example listings. In
163 schema listings, this is the default namespace and no prefix is shown.

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

166 1.2. Schema Organization and Namespaces

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

169 <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-22.xsd>

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

172 <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-22.xsd>

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

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

177 <http://www.w3.org/2000/09/xmlsig#>

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

180 **1.3. SAML Concepts (Non-Normative)**

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

184 [TBD]Need conceptual material here. Explain concepts/terms such as the domain model, SAML-
185 defined namespaces, URIs for identifiers, what is out of band/scope, extension points, etc.

186 2. SAML Assertions

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

- 189 • **Authentication:** The specified subject was authenticated by a particular means at a
190 particular time.
- 191 • **Authorization Decision:** A request to allow the specified subject to access the specified
192 object has been granted or denied.
- 193 • **Attribute:** The specified subject is associated with the supplied attributes.

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

197 2.1. Schema Header and Namespace Declarations

198 The following schema fragment defines the XML namespaces and other header information for the
199 assertion schema:

```
200 <schema  
201   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-  
202 sstc-schema-assertion-22.xsd"  
203   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
204   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
205 schema-assertion-22.xsd"  
206   xmlns="http://www.w3.org/2001/XMLSchema"  
207   elementFormDefault="unqualified">  
208   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
209     schemaLocation="xmldsig-core-schema.xsd"/>  
210   <annotation>  
211     <documentation>draft-sstc-schema-assertion-22.xsd</documentation>  
212   </annotation>  
213   ...  
214 </schema>
```

215 2.2. Simple Types

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

217 2.2.1. Simple Type IDType

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

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

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

225 The mechanism by which the application ensures that the identifier is unique is left to the
226 implementation. In the case that a pseudorandom technique is employed, the probability of two
227 randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less than
228 2^{-160} .

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

232 The following schema fragment defines the **IDType** simple type:

```
233 <simpleType name="IDType">  
234   <restriction base="string"/>  
235 </simpleType>
```

236 2.2.2. Simple Type DecisionType

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

239 Permit

240 The specified action is permitted.

241 Deny

242 The specified action is denied.

243 Indeterminate

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

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

```
246 <simpleType name="DecisionType">  
247   <restriction base="string">  
248     <enumeration value="Permit"/>  
249     <enumeration value="Deny"/>  
250     <enumeration value="Indeterminate"/>  
251   </restriction>  
252 </simpleType>
```

253 2.3. Assertions

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

255 2.3.1. Element <AssertionSpecifier>

256 The <AssertionSpecifier> element specifies an assertion either by reference or by value. It
257 contains one of the following elements:

258 <AssertionID>

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

260 <Assertion>

261 Specifies an assertion by value.

262 The following schema fragment defines the <AssertionSpecifier> element and its

263 **AssertionSpecifierType** complex type:

```
264 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>  
265 <complexType name="AssertionSpecifierType">  
266   <choice>  
267     <element ref="saml:AssertionID"/>  
268     <element ref="saml:Assertion"/>  
269   </choice>  
270 </complexType>
```

271 2.3.2. Element <AssertionID>

272 The <AssertionID> element makes a reference to a SAML assertion by means of the value the
273 assertion's `AssertionID` attribute.

274 The following schema fragment defines the <AssertionID> element:

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

276 2.3.3. Element <Assertion>

277 The <Assertion> element is of **AssertionType** complex type. This type specifies the basic
278 information that is common to all assertions, including the following elements (in order) and
279 attributes:

280 MajorVersion [Required]

281 The major version of this assertion. The identifier for the version of SAML defined in this
282 specification is 1. Processing of this attribute is specified in Section 3.5.2.

283 MinorVersion [Required]

284 The minor version of this assertion. The identifier for the version of SAML defined in this
285 specification is 0. Processing of this attribute is specified in Section 3.5.2.

286 AssertionID [Required]

287 The identifier for this assertion. It is of type **IDType**, and MUST follow the requirements
288 specified by that type for identifier uniqueness.

289 Issuer [Required]

290 The issuer of the assertion. The name of the issuer is provided as a string. The issuer
291 name SHOULD be unambiguous to the intended relying parties. SAML applications may
292 use an identifier such as a URI that is designed to be unambiguous regardless of context.

293 IssueInstant [Required]

294 The time instant of issue. It has the type **dateTime**, which is built in to the W3C XML
295 Schema Datatypes specification [**Schema2**].

296 <Conditions> [Optional]

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

298 <Advice> [Optional]

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

301 One or more of the following statement elements:

302 <Statement>

303 A statement defined in an extension schema.

304 <SubjectStatement>

305 A subject statement defined in an extension schema.

306 <AuthenticationStatement>

307 An authentication statement.

308 <AuthorizationDecisionStatement>

309 An authorization decision statement.

310 <AttributeStatement>

311 An attribute statement.

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

```
314 <complexType name="AssertionType">  
315 <sequence>  
316 <element ref="saml:Conditions" minOccurs="0"/>  
317 <element ref="saml:Advice" minOccurs="0"/>  
318 <choice minOccurs="0" maxOccurs="unbounded">  
319 <element ref="saml:Statement"/>
```

```

320         <element ref="saml:SubjectStatement" />
321         <element ref="saml:AuthenticationStatement" />
322         <element ref="saml:AuthorizationDecisionStatement" />
323         <element ref="saml:AttributeStatement" />
324     </choice>
325     <element ref="ds:Signature" minOccurs="0" maxOccurs="unbounded" />
326 </sequence>
327     <attribute name="MajorVersion" type="integer" use="required" />
328     <attribute name="MinorVersion" type="integer" use="required" />
329     <attribute name="AssertionID" type="saml:IDType" use="required" />
330     <attribute name="Issuer" type="string" use="required" />
331     <attribute name="IssueInstant" type="dateTime" use="required" />
332 </complexType>

```

333 2.3.3.1. Element <Conditions>

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

- 338 • If any condition evaluates to Invalid, the assertion status is Invalid.
- 339 • If no condition evaluates to Invalid and one or more conditions evaluate to
340 Indeterminate, the assertion status is Indeterminate.
- 341 • If no conditions are supplied or all the specified conditions evaluate to Valid, the assertion
342 status is Valid.

343 The <Conditions> element MAY be extended to contain additional conditions. If an element
344 contained within a <Conditions> element is encountered that is not understood, the status of the
345 condition MUST be evaluated to Indeterminate.

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

347 NotBefore [Optional]

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

349 NotOnOrAfter [Optional]

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

351 <Condition> [Zero or more]

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

353 <AudienceRestrictionCondition> [Any Number]

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

355 <TargetRestrictionCondition> [Any Number]

356 The <TargetRestriction> condition is used to limit the use of the assertion to a particular
357 relying party.

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

```

360     <element name="Conditions" type="saml:ConditionsType" />
361     <complexType name="ConditionsType">
362         <choice minOccurs="0" maxOccurs="unbounded">
363             <element ref="saml:Condition" />
364             <element ref="saml:AudienceRestrictionCondition" />
365             <element ref="saml:TargetRestrictionCondition" />
366         </choice>
367         <attribute name="NotBefore" type="dateTime" use="optional" />
368         <attribute name="NotOnOrAfter" type="dateTime" use="optional" />

```

369 `</complexType>`

370 **2.3.3.1.1 Attributes *NotBefore* and *NotOnOrAfter***

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

372 The *NotBefore* attribute specifies the time instant at which the validity interval begins. The
373 *NotOnOrAfter* attribute specifies the time instant at which the validity interval has ended.

374 If the value for either *NotBefore* or *NotOnOrAfter* is omitted or is equal to the start of the epoch,
375 it is considered unspecified. If the *NotBefore* attribute is unspecified (and if any other conditions
376 that are supplied evaluate to *Valid*), the assertion is valid at any time before the time instant
377 specified by the *NotOnOrAfter* attribute. If the *NotOnOrAfter* attribute is unspecified (and if any
378 other conditions that are supplied evaluate to *Valid*), the assertion is valid from the time instant
379 specified by the *NotBefore* attribute with no expiry. If neither attribute is specified (and if any other
380 conditions that are supplied evaluate to *Valid*), the assertion is valid at any time.

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

385 **2.3.3.1.2 Element *<Condition>***

386 The *<Condition>* element serves as an extension point for new conditions. Its
387 **ConditionAbstractType** complex type is abstract; extension elements **MUST** use the `xsi:type`
388 attribute to indicate the derived type.

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

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

393 **2.3.3.1.3 Elements *<AudienceRestrictionCondition>* and *<Audience>***

394 The *<AudienceRestrictionCondition>* element specifies that the assertion is addressed to
395 one or more specific audiences. Although a party that is outside the audiences specified is capable
396 of drawing conclusions from an assertion, the issuer explicitly makes no representation as to
397 accuracy or trustworthiness to such a party.

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

400 The condition evaluates to *Valid* if and only if the relying party is a member of one or more of the
401 audiences specified.

402 The issuer of an assertion cannot prevent a party to whom it is disclosed from making a decision on
403 the basis of the information provided. However, the *<AudienceRestrictionCondition>*
404 element allows the issuer to state explicitly that no warranty is provided to such a party in a
405 machine- and human-readable form. While there can be no guarantee that a court would upholding
406 such a warranty exclusion in every circumstance, the probability of upholding the warranty
407 exclusion is considerably improved.

408 The following schema fragment defines the *<AudienceRestrictionCondition>* element and
409 its **AudienceRestrictionConditionType** complex type:

```
410 <element name="AudienceRestrictionCondition"  
411   type="saml:AudienceRestrictionConditionType"/>  
412 <complexType name="AudienceRestrictionConditionType">  
413   <complexContent>
```

```

414     <extension base="saml:ConditionAbstractType">
415         <sequence>
416             <element ref="saml:Audience"
417                 minOccurs="1" maxOccurs="unbounded"/>
418         </sequence>
419     </extension>
420 </complexContent>
421 </complexType>
422 <element name="Audience" type="anyURI"/>

```

423 2.3.3.1.4 Condition Type TargetRestrictionType

424 The <TargetRestriction> element is used to limit the use of the assertion to a particular relying
425 party. This is useful to prevent malicious forwarding of assertions to unintended recipients.

426 The target is identified by a URI. The condition evaluates to true if one or more URIs identify the
427 recipient or a resource managed by the recipient.

428 The following schema fragment defines the <TargetRestrictionCondition> element and its
429 **TargetRestrictionConditionType** complex type:

```

430 <element name="TargetRestrictionCondition"
431     type="saml:TargetRestrictionConditionType"/>
432 <complexType name="TargetRestrictionConditionType">
433     <complexContent>
434         <extension base="saml:ConditionAbstractType">
435             <sequence>
436                 <element ref="saml:Target"
437                     minOccurs="1" maxOccurs="unbounded"/>
438             </sequence>
439         </extension>
440     </complexContent>
441 </complexType>
442 <element name="Target" type="anyURI"/>

```

443 2.3.3.2. Elements <Advice> and <AdviceElement>

444 The <Advice> element contains any additional information that the issuer wishes to provide. This
445 information MAY be ignored by applications without affecting either the semantics or the validity of
446 the assertion.

447 The <Advice> element contains a mixture of zero or more <AssertionSpecifier> elements,
448 <AdviceElement> elements, and elements in other namespaces, with lax schema validation in
449 effect for these other elements.

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

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

455 The following schema fragment defines the <Advice> element and its **AdviceType** complex type,
456 along with the <AdviceElement> element and its **AdviceAbstractType** complex type:

```

457 <element name="Advice" type="saml:AdviceType"/>
458 <complexType name="AdviceType">
459     <sequence>
460         <choice minOccurs="0" maxOccurs="unbounded">
461             <element ref="saml:AssertionSpecifier"/>
462             <element ref="saml:AdviceElement"/>
463             <any namespace="##other" processContents="lax"/>

```

```

464         </choice>
465     </sequence>
466 </complexType>
467 <element name="AdviceElement" type="saml:AdviceAbstractType" />
468 <complexType name="AdviceAbstractType" />

```

2.4. Statements

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

2.4.1. Element <Statement>

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

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

```

477 <element name="Statement" type="saml:StatementAbstractType" />
478 <complexType name="StatementAbstractType" abstract="true" />

```

2.4.2. Element <SubjectStatement>

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

The following schema fragment defines the <SubjectStatement> element and its **SubjectStatementAbstractType** abstract type:

```

487 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType" />
488 <complexType name="SubjectStatementAbstractType" abstract="true">
489     <complexContent>
490         <extension base="saml:StatementAbstractType">
491             <sequence>
492                 <element ref="saml:Subject" />
493             </sequence>
494         </extension>
495     </complexContent>
496 </complexType>

```

2.4.2.1. Element <Subject>

The <Subject> element specifies one or more subjects. It contains either or both of the following elements:

<NameIdentifier>

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

<SubjectConfirmation>

Information that allows the subject to be authenticated.

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

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

```
511 <element name="Subject" type="saml:SubjectType"/>
512 <complexType name="SubjectType">
513   <choice maxOccurs="unbounded">
514     <sequence>
515       <element ref="saml:NameIdentifier"/>
516       <element ref="saml:SubjectConfirmation" minOccurs="0"/>
517     </sequence>
518     <element ref="saml:SubjectConfirmation"/>
519   </choice>
520 </complexType>
```

521 2.4.2.2. Element <NameIdentifier>

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

524 SecurityDomain

525 The security domain governing the name of the subject.

526 Name

527 The name of the subject.

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

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

```
533 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
534 <complexType name="NameIdentifierType">
535   <attribute name="SecurityDomain" type="string"/>
536   <attribute name="Name" type="string"/>
537 </complexType>
```

538 2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and 539 <SubjectConfirmationData>

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

542 <ConfirmationMethod> [One or more]

543 A URI that identifies a protocol to be used to authenticate the subject. URIs identifying
544 common authentication protocols are listed in Section 7.

545 <SubjectConfirmationData> [Zero or more]

546 Additional authentication information to be used by a specific authentication protocol.

547 <ds:KeyInfo> [Optional]

548 An XML Signature [**XMLSig**] element that specifies a cryptographic key held by the
549 subject.

550 The following schema fragment defines the <SubjectConfirmation> element and its
551 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData>
552 element and the <ConfirmationMethod> element:

```
553 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
554 <complexType name="SubjectConfirmationType">
555   <sequence>
556     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
557     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
```



```

558         <element ref="ds:KeyInfo" minOccurs="0"/>
559     </sequence>
560 </complexType>
561 <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
562 <element name="ConfirmationMethod" type="anyURI"/>

```

563 2.4.3. Element <AuthenticationStatement>

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

568 AuthenticationMethod [Required]

569 A URI that specifies the type of authentication that took place. URIs identifying common
570 authentication protocols are listed in Section 7.

571 AuthenticationInstant [Required]

572 Specifies the time at which the authentication took place.

573 <AuthenticationLocality> [Optional]

574 Specifies the DNS domain name and IP address for the system entity that performed the
575 authentication.

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

```

578 <element name="AuthenticationStatement"
579     type="saml:AuthenticationStatementType"/>
580 <complexType name="AuthenticationStatementType">
581     <complexContent>
582         <extension base="saml:SubjectStatementAbstractType">
583             <sequence>
584                 <element ref="saml:AuthenticationLocality" minOccurs="0"/>
585             </sequence>
586             <attribute name="AuthenticationMethod" type="anyURI"/>
587             <attribute name="AuthenticationInstant" type="dateTime"/>
588         </extension>
589     </complexContent>
590 </complexType>

```

591 2.4.3.1. Element <AuthenticationLocality>

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

594 IPAddress [Optional]

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

596 DNSAddress [Required]

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

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

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

```

602 <element name="AuthenticationLocality"
603     type="saml:AuthenticationLocalityType"/>
604 <complexType name="AuthenticationLocalityType">
605     <attribute name="IPAddress" type="string" use="optional"/>
606     <attribute name="DNSAddress" type="string" use="optional"/>
607 </complexType>

```


608 2.4.4. Element <AuthorizationDecisionStatement>

609 The <AuthorizationDecisionStatement> element supplies a statement by the issuer that the
610 request for access by the specified subject to the specified resource has resulted in the specified
611 decision on the basis of some optionally specified evidence. It is of type
612 **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the
613 addition of the following elements (in order) and attributes:

614 Resource [Optional]

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

616 Decision [Optional]

617 The decision rendered by the issuer with respect to the specified resource. The value is of
618 the **DecisionType** simple type.

619 <Actions> [Required]

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

621 <Evidence> [Zero or more]

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

623 The following schema fragment defines the <AuthorizationDecisionStatement> element
624 and its **AuthorizationDecisionStatementType** complex type:

```
625 <element name="AuthorizationDecisionStatement"  
626 type="saml:AuthorizationDecisionStatementType"/>  
627 <complexType name="AuthorizationDecisionStatementType">  
628 <complexContent>  
629 <extension base="saml:SubjectStatementAbstractType">  
630 <sequence>  
631 <element ref="saml:Actions"/>  
632 <element ref="saml:Evidence" minOccurs="0"  
633 maxOccurs="unbounded"/>  
634 </sequence>  
635 <attribute name="Resource" type="anyURI" use="optional"/>  
636 <attribute name="Decision" type="saml:DecisionType"  
637 use="optional"/>  
638 </extension>  
639 </complexContent>  
640 </complexType>
```

641 2.4.4.1. Elements <Actions> and <Action>

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

644 Namespace [Optional]

645 A URI representing the namespace in which the names of specified actions are to be
646 interpreted. If this element is absent, the namespace [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/rwcdc-negation)
647 [open.org/committees/security/docs/draft-sstc-core-22/rwcdc-negation](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/rwcdc-negation) specified in section
648 7.2.2 is in effect by default.

649 <Action> [One or more]

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

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

```
653 <element name="Actions" type="saml:ActionTypes"/>  
654 <complexType name="ActionTypes">  
655 <sequence>  
656 <element ref="saml:Action" maxOccurs="unbounded"/>  
657 </sequence>
```

```
658     <attribute name="Namespace" type="anyURI" use="optional"/>
659 </complexType>
660 <element name="Action" type="string"/>
```

661 2.4.4.2. Element <Evidence>

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

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

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

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

670 2.4.5. Element <AttributeStatement>

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

674 <Attribute> [One or More]

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

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

```
678 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
679 <complexType name="AttributeStatementType">
680 <complexContent>
681 <extension base="saml:SubjectStatementAbstractType">
682 <sequence>
683 <element ref="saml:Attribute" maxOccurs="unbounded"/>
684 </sequence>
685 </extension>
686 </complexContent>
687 </complexType>
```

688 2.4.5.1. Elements <AttributeDesignator> and <Attribute>

689 The <AttributeDesignator> element identifies an attribute name within an attribute
690 namespace. It has the **AttributeDesignatorType** complex type. It is used in an attribute assertion
691 query to request that attribute values within a specific namespace be returned (see 3.4.4 for more
692 information). The <AttributeDesignator> element contains the following XML attributes:

693 AttributeNamespace [Required]

694 The namespace in which the AttributeName elements are interpreted.

695 AttributeName [Required]

696 The name of the attribute.

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

```
699 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
700 <complexType name="AttributeDesignatorType">
701 <attribute name="AttributeName" type="string"/>
702 <attribute name="AttributeNamespace" type="anyURI"/>
703 </complexType>
```

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

707 <AttributeValue> [Required]
708 The value of the attribute.

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

```
711 <element name="Attribute" type="saml:AttributeType"/>  
712 <complexType name="AttributeType">  
713 <complexContent>  
714 <extension base="saml:AttributeDesignatorType">  
715 <sequence>  
716 <element ref="saml:AttributeValue"/>  
717 </sequence>  
718 </extension>  
719 </complexContent>  
720 </complexType>
```

721 **2.4.5.1.1 Element <AttributeValue>**

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

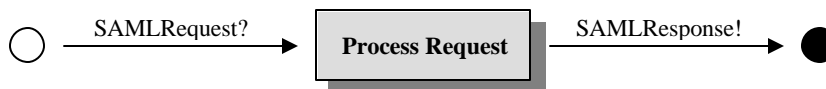
725 The following schema fragment defines the <AttributeValue> element and its
726 **AttributeValueType** complex type:

```
727 <element name="AttributeValue" type="saml:AttributeValueType"/>  
728 <complexType name="AttributeValueType">  
729 <sequence>  
730 <any namespace="##any" processContents="lax"  
731 minOccurs="0" maxOccurs="unbounded"/>  
732 </sequence>  
733 </complexType>
```

734 3. SAML Protocol

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

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



741

742

Figure 1: SAML Request-Response Protocol

743 3.1. Schema Header and Namespace Declarations

744 The following schema fragment defines the XML namespaces and other header information for the
745 protocol schema:

```
746 <schema  
747   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-  
748 sstc-schema-protocol-22.xsd"  
749   xmlns="http://www.w3.org/2001/XMLSchema"  
750   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
751 schema-protocol-22.xsd"  
752   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
753 schema-assertion-22.xsd"  
754   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
755   elementFormDefault="unqualified">  
756   <import namespace="http://www.oasis-open.org/committees/security/docs/draft-  
757 sstc-schema-assertion-22.xsd"  
758     schemaLocation="draft-sstc-schema-assertion-22.xsd" />  
759   <import namespace="http://www.w3.org/2000/09/xmldsig#" />  
760     schemaLocation="xmldsig-core-schema.xsd" />  
761   <annotation>  
762     <documentation>draft-sstc-schema-protocol-22.xsd</documentation>  
763   </annotation>  
764   ...  
765 </schema>
```

766 3.2. Simple Types

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

768 3.2.1. Simple Type StatusCodeType

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

771 Success

772 The request succeeded.

773 Failure

774 The request could not be performed by the service.

775 Error

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

777 Unknown
778 The request failed for unknown reasons.

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

```
780 <simpleType name="StatusCodeType">  
781 <restriction base="string">  
782 <enumeration value="Success"/>  
783 <enumeration value="Failure"/>  
784 <enumeration value="Error"/>  
785 <enumeration value="Unknown"/>  
786 </restriction>  
787 </simpleType>
```

788 3.3. Requests

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

790 3.3.1. Complex Type RequestAbstractType

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

793 RequestID [Required]

794 An identifier for the request. It is of type **IDType**, and MUST follow the requirements
795 specified by that type for identifier uniqueness. The values of the RequestID attribute in a
796 request and the InResponseTo attribute in the corresponding response MUST match.

797 MajorVersion [Required]

798 The major version of this request. The identifier for the version of SAML defined in this
799 specification is 1. Processing of this attribute is specified in Section 3.5.2.

800 MinorVersion [Required]

801 The minor version of this request. The identifier for the version of SAML defined in this
802 specification is 0. Processing of this attribute is specified in Section 3.5.2.

803 <RespondWith> [Any Number]

804 Each <RespondWith> element specifies a type of response that is acceptable to the
805 requestor.

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

```
807 <complexType name="RequestAbstractType" abstract="true">  
808 <sequence>  
809 <element ref="samlp:RespondWith"  
810 minOccurs="0" maxOccurs="unbounded"/>  
811 <element ref="ds:Signature" minOccurs="0" maxOccurs="unbounded"/>  
812 </sequence>  
813 <attribute name="RequestID" type="saml:IDType" use="required"/>  
814 <attribute name="MajorVersion" type="integer" use="required"/>  
815 <attribute name="MinorVersion" type="integer" use="required"/>  
816 </complexType>
```

817 3.3.1.1. Element <RespondWith>

818 The <RespondWith> element specifies a type of response that is acceptable to the requestor. If
819 no <RespondWith> element is specified the default is SingleStatement. Acceptable values for
820 the <RespondWith> element are:

821 SingleStatement

822 An assertion carrying exactly one statement element.

- 823 MultipleStatement
- 824 An assertion carrying at least one statement element.
- 825 AuthenticationStatement
- 826 An assertion carrying an Authentication statement.
- 827 AuthorizationDecisionStatement
- 828 An assertion carrying an Authorization Decision statement.
- 829 AttributeStatement
- 830 An assertion carrying an Attribute statement.
- 831 *Schema URI*
- 832 An assertion containing additional elements from the specified schema.

833 The following schema fragment defines the `<RespondWith>` element:

```
834 <element name="RespondWith" type="anyURI"/>
```

835 3.3.2. Element `<Request>`

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

- 840 `<Query>`
- 841 An extension point that allows extension schemas to define new types of query.
- 842 `<SubjectQuery>`
- 843 An extension point that allows extension schemas to define new types of query that specify
 844 a single SAML subject.
- 845 `<AuthenticationQuery>`
- 846 Makes a query for authentication information.
- 847 `<AttributeQuery>`
- 848 Makes a query for attribute information.
- 849 `<AuthorizationDecisionQuery>`
- 850 Makes a query for an authorization decision.
- 851 `<AssertionID>` [One or more]
- 852 Requests an assertion by reference to its assertion identifier.
- 853 `<AssertionArtifact>` [One or more]
- 854 Requests an assertion by supplying an assertion artifact that represents it.

855 The following schema fragment defines the `<Request>` element and its **RequestType** complex
 856 type:

```
857 <element name="Request" type="samlp:RequestType"/>
858 <complexType name="RequestType">
859   <complexContent>
860     <extension base="samlp:RequestAbstractType">
861       <choice>
862         <element ref="samlp:Query"/>
863         <element ref="samlp:SubjectQuery"/>
864         <element ref="samlp:AuthenticationQuery"/>
865         <element ref="samlp:AttributeQuery"/>
866         <element ref="samlp:AuthorizationDecisionQuery"/>
867         <element ref="saml:AssertionID" maxOccurs="unbounded"/>
868         <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
869       </choice>
870     </extension>
```

```
871     </complexContent>
872   </complexType>
873   <element name="AssertionArtifact" type="string"/>
```

874 3.4. Queries

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

876 3.4.1. Element <Query>

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

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

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

885 3.4.2. Element <SubjectQuery>

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

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

```
892   <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
893   <complexType name="SubjectQueryAbstractType" abstract="true">
894     <complexContent>
895       <extension base="samlp:QueryAbstractType">
896         <sequence>
897           <element ref="saml:Subject"/>
898         </sequence>
899       </extension>
900     </complexContent>
901   </complexType>
```

902 3.4.3. Element <AuthenticationQuery>

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

```
907 <ConfirmationMethod> [Optional]
908     A filter for possible responses. If it is present, the query made is “What authentication
909     assertions do you have for this subject with the supplied confirmation method?”
```

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

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

```
917 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
918 <complexType name="AuthenticationQueryType">
919   <complexContent>
920     <extension base="samlp:SubjectQueryAbstractType">
921       <sequence>
922         <element ref="saml:ConfirmationMethod" minOccurs="0"/>
923       </sequence>
924     </extension>
925   </complexContent>
926 </complexType>
```

927 3.4.4. Element <AttributeQuery>

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

932 <AttributeDesignator> [Zero or more] (see Section 2.4.5.1)
933 Each <AttributeDesignator> element specifies an attribute whose value is to be
934 returned. If no attributes are specified, the list of desired attributes is implicit and
935 application-specific.

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

```
938 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
939 <complexType name="AttributeQueryType">
940   <complexContent>
941     <extension base="samlp:SubjectQueryAbstractType">
942       <sequence>
943         <element ref="saml:AttributeDesignator"
944           minOccurs="0" maxOccurs="unbounded"/>
945       </sequence>
946       <attribute name="CompletenessSpecifier"
947         type="samlp:CompletenessSpecifierType" use="required"/>
948     </extension>
949   </complexContent>
950 </complexType>
```

951 3.4.5. Element <AuthorizationDecisionQuery>

952 The <AuthorizationDecisionQuery> element is used to make the query “Should these
953 actions on this resource be allowed for this subject, given this evidence?” The response will be in
954 the form of an assertion containing an authorization decision statement. This element is of type
955 **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition
956 of the following elements and attribute:

957 Resource [Required]
958 A URI indicating the resource for which authorization is requested.

959 <Actions> [Required]
960 The actions for which authorization is requested.

961 <Evidence> [Zero or more]
962 An assertion that the responder MAY rely on in making its response.

963 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
964 **AuthorizationDecisionQueryType** complex type:


```

965     <element name="AuthorizationDecisionQuery"
966 type="samlp:AuthorizationDecisionQueryType" />
967     <complexType name="AuthorizationDecisionQueryType">
968       <complexContent>
969         <extension base="samlp:SubjectQueryAbstractType">
970           <sequence>
971             <element ref="saml:Actions" />
972             <element ref="saml:Evidence"
973               minOccurs="0" maxOccurs="unbounded" />
974           </sequence>
975           <attribute name="Resource" type="anyURI" />
976         </extension>
977       </complexContent>
978     </complexType>

```

3.5. Responses

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

3.5.1. Complex Type ResponseAbstractType

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

ResponseID [Required]

An identifier for the response. It is of type **IDType**, and MUST follow the requirements specified by that type for identifier uniqueness.

InResponseTo [Required]

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

MajorVersion [Required]

The major version of this response. The identifier for the version of SAML defined in this specification is 1. Processing of this attribute is specified in Section 3.5.2.

MinorVersion [Required]

The minor version of this response. The identifier for the version of SAML defined in this specification is 0. Processing of this attribute is specified in Section 3.5.2.

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

```

997     <complexType name="ResponseAbstractType" abstract="true">
998       <sequence>
999         <element ref="ds:Signature" minOccurs="0" maxOccurs="unbounded" />
1000       </sequence>
1001       <attribute name="ResponseID" type="saml:IDType" use="required" />
1002       <attribute name="InResponseTo" type="saml:IDType" use="required" />
1003       <attribute name="MajorVersion" type="integer" use="required" />
1004       <attribute name="MinorVersion" type="integer" use="required" />
1005     </complexType>

```

3.5.2. Element <Response>

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

StatusCode [Required] (see Section 3.2.1)

A code representing the status of the corresponding request.

1013 <Assertion> (see Section 2.3.3)
 1014 Specifies an assertion by value.
 1015 <SingleAssertion>
 1016 Specifies an assertion containing a single statement by value.
 1017 <MultipleAssertion>
 1018 Specifies an assertion containing multiple statements by value.
 1019 The following schema fragment defines the <Response> element and its **ResponseType** complex
 1020 type:

```

1021 <element name="Response" type="samlp:ResponseType" />
1022 <complexType name="ResponseType">
1023   <complexContent>
1024     <extension base="samlp:ResponseAbstractType">
1025       <sequence>
1026         <element ref="samlp:StatusReason"
1027           minOccurs="0" maxOccurs="unbounded" />
1028         <element ref="saml:Assertion"
1029           minOccurs="0" maxOccurs="unbounded" />
1030       </sequence>
1031       <attribute name="StatusCode"
1032         type="samlp:StatusCodeType" use="required" />
1033     </extension>
1034   </complexContent>
1035 </complexType>

```

1036 3.5.2.1. Element <StatusReason>

1037 The <StatusReason> element provides additional information that indicates the reason for the
 1038 return of an `Error` or `Failure` status code. The following values are defined. Implementations
 1039 MAY define additional codes:

1040 `RequestVersionTooHigh`

1041 The protocol version specified in the request is a major upgrade from the highest protocol
 1042 version supported by the responder.

1043 `RequestVersionTooLow`

1044 The responder cannot respond to the particular request using the SAML version specified
 1045 in the request because it is too low.

1046 `RequestVersionDeprecated`

1047 The responder does not respond to any requests with the protocol version specified in the
 1048 request.

1049 `TooManyResponses`

1050 The response would contain more elements than the responder will return.

1051 The following schema fragment defines the <StatusReason> element:

```

1052 <element name="StatusReason" type="string" />

```

4. SAML Versioning

1053

1054 SAML version information appears in the following elements:

- 1055 • <Assertion>
- 1056 • <Request>
- 1057 • <Response>

1058 The version numbering of the SAML assertion is independent of the version number of the SAML
1059 request-response protocol. The version information for each consists of a major version number
1060 and a minor version number, both of which are integers. In accordance with industry practice a
1061 version number SHOULD be presented to the user in the form *Major.Minor*. This document defines
1062 SAML Assertions 1.0 and SAML Protocol 1.0.

1063 The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

$$1064 \quad Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B = Minor_A)$$

1065 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that
1066 are the same as or higher than the corresponding version number in the SAML version that
1067 immediately preceded it.

1068 New versions of SAML SHALL assign new version numbers as follows:

- 1069 • **Documentation change:** $(Major_B = Major_A) \wedge (Minor_B = Minor_A)$
1070 If the major and minor version numbers are unchanged, the new version *B* only introduces
1071 changes to the documentation that raise no compatibility issues with an implementation of
1072 version *A*.
- 1073 • **Minor upgrade:** $(Major_B = Major_A) \wedge (Minor_B > Minor_A)$
1074 If the major version number of versions *A* and *B* are the same and the minor version
1075 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the
1076 SAML schema and semantics but any changes that are introduced in *B* SHALL be
1077 compatible with version *A*.
- 1078 • **Major upgrade:** $Major_B > Major_A$
1079 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY
1080 introduce changes to the SAML schema and semantics that are incompatible with *A*.

4.1. Assertion Version

1081

1082 A SAML application MUST NOT issue any assertion whose version number is not supported.

1083 A SAML application MUST reject any assertion whose major version number is not supported.

1084 A SAML application MAY reject any assertion whose version number is higher than the highest
1085 supported version.

4.2. Request Version

1086

1087 A SAML application SHOULD issue requests that specify the highest SAML version supported by
1088 both the sender and recipient.

1089 If the SAML application does not know the capabilities of the recipient it should assume that it
1090 supports the highest SAML version supported by the sender.

1091 **4.3. Response Version**

1092 A SAML application MUST NOT issue responses that specify a higher SAML version number than
1093 the corresponding request.

1094 A SAML application MUST NOT issue a response that has a major version number that is lower
1095 than the major version number of the corresponding request except to report the error
1096 `RequestVersionTooHigh`.

1097 Incompatible protocol versions MAY cause the following errors to be reported:

1098 `RequestVersionTooHigh`

1099 The protocol version specified in the request is a major upgrade from the highest protocol
1100 version supported by the responder.

1101 `RequestVersionTooLow`

1102 The responder cannot respond to the particular request using the SAML version specified
1103 in the request because it is too low.

1104 `RequestVersionDeprecated`

1105 The responder does not respond to any requests with the protocol version specified in the
1106 request.

5. SAML & XML-Signature Syntax and Processing

SAML Assertions, Request and Response messages may be signed, with the following benefits:

- An Assertion signed by the issuer (AP). This supports :
 - (1) Message integrity
 - (2) Authentication of the issuer to a relying party
 - (3) If the signature is based on the issuer's public-private key pair, then it also provides for non-repudiation of origin.
- A SAML request or a SAML response message signed by the message originator. This supports :
 - (1) Message integrity
 - (2) Authentication of message origin to a destination
 - (3) If the signature is based on the originator's public-private key pair, then it also provides for non-repudiation of origin.

Note :

- SAML documents may be the subject of signatures from in many different packaging contexts. [SIG] provides a framework for signing in XML and is the framework of choice. However, signing may also take place in the context of S/MIME or Java objects that contain SAML documents. One goal is to ensure compatibility with this type of "foreign" digital signing.
- It is useful to characterize situations when a digital signature is NOT required in SAML.
 - (1) Assertions: asserting party has provided the assertion to the relying party, authenticated by means other than digital signature and the channel is secure. In other words, the RP has obtained the assertion from the AP directly (no intermediaries) thru a secure channel and the AP has authenticated to the RP.
 - (2) Request/Response messages: the originator has authenticated to the destination and the destination has obtained the assertion directly from the originator (no intermediaries) thru secure channel(s).

Many different techniques are available for "direct" authentication and secure channel between two parties. The list includes SSL, HMAC, password-based login etc. Also the security requirement depends on the communicating applications and the nature of the assertion transported.
- All other contexts require the use of digital signature for assertions and request and response messages. Specifically:

- 1153 (1) An assertion obtained by a relying party from an entity other than the asserting
1154 party MUST be signed by the issuer.
1155
1156 (2) SAML message obtained arriving at a destination from an entity other than the
1157 originating site MUST be signed by the origin site.
1158

1159 **5.1. Signing Assertions**

1160 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion
1161 schema – Section 2.3.3.
1162

1163 **5.2. Request/Response Signing**

1164 All SAML requests and responses MAY be signed using the XML Signature. This is reflected in the
1165 schema – Section 3.3.1 & 3.5.1.

1166 **5.3. Signature Inheritance (a.k.a. super-signatures & sub- 1167 messages)**

1168 **5.3.1. Rationale**

1169
1170 SAML assertions may be embedded within request or response messages or other XML
1171 messages, which may be signed. Request or response messages may themselves be contained
1172 within other messages that are based on other XML messaging frameworks (e.g., SOAP) and the
1173 composite object may be the subject of a signature. Another possibility is that SAML assertions or
1174 request/response messages are embedded within a non-XML messaging object (e.g., MIME
1175 package) and signed.

1176
1177 In such a case, the SAML sub-message (Assertion, request, response) may be viewed as
1178 inheriting a signature from the "super-signature" over the enclosing object, provided certain
1179 constraints are met.

- 1180
1181 (1) An assertion may be viewed as inheriting a signature from a super signature, if the super
1182 signature applies all the elements within the assertion.
1183
1184 (2) A SAML request or response may be viewed as inheriting a signature from a super
1185 signature, if the super signature applies to all the elements within the response.
1186

1187 **5.3.2. Rules for SAML Signature Inheritance**

1188
1189 Signature inheritance: occurs when SAML message (assertion/request/response) is not signed
1190 but is enclosed within signed SAML such that the signature applies to all of the elements within the
1191 message. In such a case, the SAML message is said to inherit the signature and may be
1192 considered equivalent to the case where it is explicitly signed. The SAML message inherits the
1193 "closest enclosing signature".
1194

1195 But if SAML messages need to be passed around by themselves, or embedded in other
1196 messages, they would need to be signed as per section 2.1

1197

1198 **5.4. XML Signature Profile**

1199

1200 The [SIG] specification calls out a general XML syntax for signing data with many flexibilities
1201 and choices. This section details the constraints on these facilities so that SAML processors do not
1202 have to deal with the full generality of [SIG] processing.

1203 **5.4.1. Signing formats**

1204

1205 XML Signature has three ways of representing signature in a document viz: enveloping,
1206 enveloped and detached.

1207 SAML assertions and protocols MUST use the enveloped signatures for signing assertions.

1208 **5.4.2. CanonicalizationMethod**

1209 [Sig] REQUIRES the Canonical XML (omits comments) ([http://www.w3.org/TR/2001/REC-xml-](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)
1210 [c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)). SAML implementations SHOULD use Canonical XML with no comments.

1211 **5.4.3. Transforms**

1212 [Sig] REQUIRES the enveloped signature transform
1213 <http://www.w3.org/2000/09/xmlsig#enveloped-signature>

1214 **5.4.4. KeyInfo**

1215 SAML does not restrict or impose any restrictions in this area. Therefore following [SIG]
1216 keyInfo may be absent.

1217 **5.4.5. Binding between statements in a multi-statement assertion**

1218 Use of signing does not affect semantics of statements within assertions in any way, as stated
1219 in this document Sections 1 thru 4.

1220 **5.4.6. Security considerations**

1221 **5.4.6.1. Replay Attack**

1222 The mechanisms stated here-in does not offer any counter measures against a replay attack.
1223 Other mechanisms like sequence numbers, time stamps, expiration et al need to be explored
1224 to prevent a replay attack.

1225

1226

1227

6. SAML Extensions

1228 The SAML schemas support extensibility. An example of an application that extends SAML
1229 assertions is the XTAML system for management of embedded trust roots **[XTAML]**. The following
1230 sections explain how to use the extensibility features in SAML to create extension schemas.

1231 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML
1232 elements MAY serve as the head element of a substitution group. Also, types are not defined as
1233 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss
1234 only elements that have been specifically designed to support extensibility.

1235

6.1. Assertion Schema Extension

1236 The SAML assertion schema is designed to permit separate processing of the assertion package
1237 and the statements it contains, if the extension mechanism is used for either part.

1238 The following elements are intended specifically for use as extension points in an extension
1239 schema; their types are set to *abstract*, so that the use of an `xsi:type` attribute with these
1240 elements is REQUIRED:

- 1241 • `<Assertion>`
- 1242 • `<Condition>`
- 1243 • `<Statement>`
- 1244 • `<SubjectStatement>`
- 1245 • `<AdviceElement>`

1246 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1247 • `<SingleAssertion>`
- 1248 • `<MultipleAssertion>`
- 1249 • `<AuthenticationStatement>`
- 1250 • `<AuthorizationDecisionStatement>`
- 1251 • `<AttributeStatement>`
- 1252 • `<AudienceRestrictionCondition>`

1253 Finally, the following elements are defined to allow elements from arbitrary namespaces within
1254 them, which serves as a built-in extension point without requiring an extension schema:

- 1255 • `<AttributeValue>`
- 1256 • `<Advice>`

1257

1258

6.2. Protocol Schema Extension

1259 The following elements are intended specifically for use as extension points in an extension
1260 schema; their types are set to *abstract*, so that the use of an `xsi:type` attribute with these
1261 elements is REQUIRED:

- 1262 • `<Query>`

1263 • <SubjectQuery>

1264 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1265 • <Request>

1266 • <AuthenticationQuery>

1267 • <AuthorizationDecisionQuery>

1268 • <AttributeQuery>

1269 • <Response>

1270 **6.3. Use of Type Derivation and Substitution Groups**

1271 W3C XML Schema [**Schema1**] provides two principal mechanisms for specifying an element of an
1272 extended type: type derivation and substitution groups.

1273 For example, a <Statement> element can be assigned the type **NewStatementType** by means of
1274 the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be
1275 derived from **StatementType**. The following example of a SAML assertion assumes that the
1276 extension schema (represented by the `new:` prefix) has defined this new type:

```
1277 <saml:Assertion ...>  
1278   <saml:Statement xsi:type="new:NewStatementType">  
1279     ...  
1280   </saml:Statement>  
1281 </saml:Assertion>
```

1282 Alternatively, the extension schema can define a <NewStatement> element that is a member of a
1283 substitution group that has <Statement> as a head element. For the substituted element to be
1284 schema-valid, it needs to have a type that matches or is derived from the head element's type. The
1285 following is an example of an extension schema fragment that defines this new element:

```
1286 <xsd:element "NewStatement" type="new:NewStatementType"  
1287   substitutionGroup="saml:Statement" />
```

1288 The substitution group declaration allows the <NewStatement> element to be used anywhere the
1289 SAML <Statement> element can be used. The following is an example of a SAML assertion that
1290 uses the extension element:

```
1291 <saml:Assertion ...>  
1292   <new:NewStatement>  
1293     ...  
1294   </new:NewStatement>  
1295 </saml:Assertion>
```

1296 The choice of extension method has no effect on the semantics of the XML document but does
1297 have implications for interoperability.

1298 The advantages of type derivation are as follows:

- 1299 • A document can be more fully interpreted by a parser that does not have access to the
1300 extension schema because a “native” SAML element is available.
- 1301 • At the time of writing, some W3C XML Schema validators do not support substitution
1302 groups, whereas the `xsi:type` attribute is widely supported.

1303 The advantage of substitution groups is that a document can be explained without the need to
1304 explain the functioning of the `xsi:type` attribute.

1305 **7. SAML-Defined Identifiers**

1306 The following sections define URI-based identifiers for common authentication protocols and
1307 actions.

1308 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the
1309 URN of the most current RFC that specifies the protocol is used. URIs created specifically for
1310 SAML have the initial stem:

1311 <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/>

1312 **7.1. Confirmation Method Identifiers**

1313 The following identifiers MAY be used in the <ConfirmationMethod> element (see Section
1314 2.4.2.3) to refer to common authentication protocols.

1315 **7.1.1. SAML Artifact:**

1316 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/artifact>

1317 <SubjectConfirmationData>: *Base64 (Artifact)*

1318 The subject of the assertion is the party that can present the SAML Artifact value specified in
1319 <SubjectConfirmationData>.

1320 **7.1.2. SAML Artifact (SHA-1):**

1321 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/artifact>

1322 <SubjectConfirmationData>: *Base64 (SHA1 (Artifact))*

1323 The subject of the assertion is the party that can present a SAML Artifact such that the SHA1 digest
1324 of the specified artifact matches the value specified in <SubjectConfirmationData>.

1325 **7.1.3. Holder of Key:**

1326 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/Holder-Of-Key>

1327 <ds:KeyInfo>: Any cryptographic key

1328 The subject of the assertion is the party that can demonstrate that it is the holder of the private
1329 component of the key specified in <ds:KeyInfo>.

1330 **7.1.4. Sender Vouches:**

1331 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/sender-vouches>

1332 Indicates that no other information is available about the context of use of the assertion. The
1333 Relying party SHOULD utilize other means to determine if it should process the assertion further.

1334 **7.1.5. Password (Pass-Through):**

1335 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/password>

1336 <SubjectConfirmationData>: *Base64 (Password)*

1337 The subject of the assertion is the party that can present the password value specified in
1338 <SubjectConfirmationData>.

1339 The username of the subject is specified by means of the <NameIdentifier> element.

1340 **7.1.6. Password (One-Way-Function SHA-1):**

1341 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/password-sha1>

1342 <SubjectConfirmationData>: *Base64 (SHA1 (Password))*

1343 The subject of the assertion is the party that can present the password such that the SHA1 digest of
1344 the specified password matches the value specified in <SubjectConfirmationData>.

1345 The username of the subject is specified by means of the <NameIdentifier> element.

1346 **7.1.7. Kerberos [Kerberos]**

1347 **URI:** <urn:ietf:rfc:1510>

1348 <SubjectConfirmationData>: A Kerberos Ticket

1349 **7.1.8. SSL/TLS Certificate Based Client Authentication:**

1350 **URI:** <urn:ietf:rfc:2246>

1351 <ds:KeyInfo>: Any cryptographic key

1352 **7.1.9. Object Authenticator (SHA-1):**

1353 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/object-sha1>

1354 <SubjectConfirmationData>: *Base64 (SHA1 (Object))*

1355 This authenticator element is the result of computing a digest, using the SHA-1 hash algorithm. It is
1356 used when the subject can be represented as a binary string, for example when it is an XML
1357 document or the disk image of executable code. Any preprocessing of the subject prior to
1358 computation of the digest is out of scope. The name of the subject should be conveyed in an
1359 accompanying NameIdentifier element.

1360 **7.1.10. PKCS#7**

1361 **URI:** <urn:ietf:rfc:2315>

1362 <SubjectConfirmationData>: *Base64 (PKCS#7 (Object))*

1363 This authenticator element is signed data in PKCS#7 format [PKCS#7]. The posited identity of the
1364 signer must be conveyed in an accompanying NameIdentifier element. This subject type may be
1365 included in the subject field of an authentication query, in which case the corresponding response
1366 indicates whether the posited signer is, indeed, the signer. It may be included in an attribute query,
1367 in which case, the requested attribute values for the subject authenticated by the signed data are
1368 returned. It may be included in an authorization query, in which case, the access request
1369 represented by the signed data shall be identified by the accompanying object element, and the
1370 corresponding authorization decision assertion indicates whether the signer is authorized for the
1371 access request represented by the object element.

1372 **7.1.11. Cryptographic Message Syntax**

1373 **URI:** urn:ietf:rfc:2630

1374 <SubjectConfirmationData>: *Base64* (CMS (*Object*))

1375 This authenticator element is signed data in CMS format [CMS]. See also 7.1.10

1376 **7.1.12. XML Digital Signature**

1377 **URI:** urn:ietf:rfc:2630

1378 <SubjectConfirmationData>: *Base64* (XML-SIG (*Object*))

1379 <ds:KeyInfo>: A cryptographic signing key

1380 This authenticator element is signed data in XML Signature format. See also 7.1.10

1381 **7.2. Action Namespace Identifiers**

1382 The following identifiers MAY be used in the `ActionNamespace` attribute (see Section 2.4.4.1) to
1383 refer to common sets of actions to perform on resources.

1384 **7.2.1. Read/Write/Execute/Delete/Control:**

1385 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/rwcdc>

1386 Defined actions:

1387 `Read Write Execute Delete Control`

1388 These actions are interpreted in the normal manner, i.e.

1389 `Read`

1390 `The subject may read the resource`

1391 `Write`

1392 `The subject may modify the resource`

1393 `Execute`

1394 `The subject may execute the resource`

1395 `Delete`

1396 `The subject may delete the resource`

1397 `Control`

1398 `The subject may specify the access control policy for the resource`

1399 **7.2.2. Read/Write/Execute/Delete/Control with Negation:**

1400 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/rwcdc-negation>

1401 Defined actions:

1402 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

1403 The actions specified in section 7.2.1 are interpreted in the same manner described there. Actions
1404 prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated
1405 permission is denied. Thus a subject described as being authorized to perform the action ~Read is
1406 affirmatively denied read permission.

1407 An application MUST NOT authorize both an action and its negated form.

1408 **7.2.3. Get/Head/Put/Post:**

1409 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/ghpp>

1410 Defined actions:

1411 GET HEAD PUT POST

1412 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform the GET action on a resource is authorized to retrieve it.

1414 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT
1415 and POST actions to the write permission. The correspondence is not exact however since a HTTP
1416 GET operation may cause data to be modified and a POST operation may cause modification to a
1417 resource other than the one specified in the request. For this reason a separate Action URI
1418 specifier is provided.

1419 **7.2.4. UNIX File Permissions:**

1420 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/unix>

1421 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)
1422 notation.

1423 The action string is a four digit numeric code:

1424 *extended user group world*

1425 Where the *extended* access permission has the value

1426 +2 if sgid is set

1427 +4 if suid is set

1428 The *user group* and *world* access permissions have the value

1429 +1 if execute permission is granted

1430 +2 if write permission is granted

1431 +4 if read permission is granted

1432 For example 0754 denotes the UNIX file access permission: user read, write and execute, group
1433 read and execute and world read.

8. SAML Schema Listings

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

8.1. Assertion Schema

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
  targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-22.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema" xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-22.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified">
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd"/>
  <annotation>
    <documentation>draft-sstc-schema-assertion-22.xsd</documentation>
  </annotation>
  <simpleType name="IDType">
    <restriction base="string"/>
  </simpleType>
  <simpleType name="DecisionType">
    <restriction base="string">
      <enumeration value="Permit"/>
      <enumeration value="Deny"/>
      <enumeration value="Indeterminate"/>
    </restriction>
  </simpleType>
  <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
  <complexType name="AssertionSpecifierType">
    <choice>
      <element ref="saml:AssertionID"/>
      <element ref="saml:Assertion"/>
    </choice>
  </complexType>
  <element name="AssertionID" type="saml:IDType"/>
  <element name="Assertion" type="saml:AssertionType"/>
  <complexType name="AssertionType">
    <sequence>
      <element ref="saml:Conditions" minOccurs="0"/>
      <element ref="saml:Advice" minOccurs="0"/>
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="saml:Statement"/>
        <element ref="saml:SubjectStatement"/>
        <element ref="saml:AuthenticationStatement"/>
        <element ref="saml:AuthorizationDecisionStatement"/>
        <element ref="saml:AttributeStatement"/>
      </choice>
      <element ref="ds:Signature" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="MajorVersion" type="integer" use="required"/>
    <attribute name="MinorVersion" type="integer" use="required"/>
    <attribute name="AssertionID" type="saml:IDType" use="required"/>
    <attribute name="Issuer" type="string" use="required"/>
    <attribute name="IssueInstant" type="dateTime" use="required"/>
  </complexType>
```

```

1491 <element name="Conditions" type="saml:ConditionsType" />
1492 <complexType name="ConditionsType">
1493   <choice minOccurs="0" maxOccurs="unbounded">
1494     <element ref="saml:Condition" />
1495     <element ref="saml:AudienceRestrictionCondition" />
1496   </choice>
1497   <attribute name="NotBefore" type="dateTime" use="optional" />
1498   <attribute name="NotOnOrAfter" type="dateTime" use="optional" />
1499 </complexType>
1500 <element name="Condition" type="saml:ConditionAbstractType" />
1501 <complexType name="ConditionAbstractType" abstract="true" />
1502 <element name="AudienceRestrictionCondition"
1503   type="saml:AudienceRestrictionConditionType" />
1504 <complexType name="AudienceRestrictionConditionType">
1505   <complexContent>
1506     <extension base="saml:ConditionAbstractType">
1507       <sequence>
1508         <element ref="saml:Audience" maxOccurs="unbounded" />
1509       </sequence>
1510     </extension>
1511   </complexContent>
1512 </complexType>
1513 <element name="Audience" type="anyURI" />
1514 <element name="TargetRestrictionCondition"
1515   type="saml:TargetRestrictionConditionType" />
1516 <complexType name="TargetRestrictionConditionType">
1517   <complexContent>
1518     <extension base="saml:ConditionAbstractType">
1519       <sequence>
1520         <element ref="saml:Target"
1521           minOccurs="1" maxOccurs="unbounded" />
1522       </sequence>
1523     </extension>
1524   </complexContent>
1525 </complexType>
1526 <element name="Target" type="anyURI" />
1527 <element name="Advice" type="saml:AdviceType" />
1528 <complexType name="AdviceType">
1529   <sequence>
1530     <choice minOccurs="0" maxOccurs="unbounded">
1531       <element ref="saml:AssertionSpecifier" />
1532       <element ref="saml:AdviceElement" />
1533       <any namespace="##other" processContents="lax" />
1534     </choice>
1535   </sequence>
1536 </complexType>
1537 <element name="AdviceElement" type="saml:AdviceAbstractType" />
1538 <complexType name="AdviceAbstractType" />
1539 <element name="Statement" type="saml:StatementAbstractType" />
1540 <complexType name="StatementAbstractType" abstract="true" />
1541 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType" />
1542 <complexType name="SubjectStatementAbstractType" abstract="true">
1543   <complexContent>
1544     <extension base="saml:StatementAbstractType">
1545       <sequence>
1546         <element ref="saml:Subject" />
1547       </sequence>
1548     </extension>
1549   </complexContent>
1550 </complexType>
1551 <element name="Subject" type="saml:SubjectType" />
1552 <complexType name="SubjectType">
1553   <choice maxOccurs="unbounded">

```

```

1554     <sequence>
1555         <element ref="saml:NameIdentifier"/>
1556         <element ref="saml:SubjectConfirmation" minOccurs="0"/>
1557     </sequence>
1558     <element ref="saml:SubjectConfirmation"/>
1559 </choice>
1560 </complexType>
1561 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1562 <complexType name="NameIdentifierType">
1563     <attribute name="SecurityDomain" type="string"/>
1564     <attribute name="Name" type="string"/>
1565 </complexType>
1566 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1567 <complexType name="SubjectConfirmationType">
1568     <sequence>
1569         <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1570         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
1571         <element ref="ds:KeyInfo" minOccurs="0"/>
1572     </sequence>
1573 </complexType>
1574 <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
1575 <element name="ConfirmationMethod" type="anyURI"/>
1576 <element name="AuthenticationStatement"
1577     type="saml:AuthenticationStatementType"/>
1578 <complexType name="AuthenticationStatementType">
1579     <complexContent>
1580         <extension base="saml:SubjectStatementAbstractType">
1581             <sequence>
1582                 <element ref="saml:AuthenticationLocality" minOccurs="0"/>
1583             </sequence>
1584             <attribute name="AuthenticationMethod" type="anyURI"/>
1585             <attribute name="AuthenticationInstant" type="dateTime"/>
1586         </extension>
1587     </complexContent>
1588 </complexType>
1589 <element name="AuthenticationLocality"
1590     type="saml:AuthenticationLocalityType"/>
1591 <complexType name="AuthenticationLocalityType">
1592     <attribute name="IPAddress" type="string" use="optional"/>
1593     <attribute name="DNSAddress" type="string" use="optional"/>
1594 </complexType>
1595 <element name="AuthorizationDecisionStatement"
1596     type="saml:AuthorizationDecisionStatementType"/>
1597 <complexType name="AuthorizationDecisionStatementType">
1598     <complexContent>
1599         <extension base="saml:SubjectStatementAbstractType">
1600             <sequence>
1601                 <element ref="saml:Actions"/>
1602                 <element ref="saml:Evidence"
1603                     minOccurs="0" maxOccurs="unbounded"/>
1604             </sequence>
1605             <attribute name="Resource" type="anyURI" use="optional"/>
1606             <attribute name="Decision"
1607                 type="saml:DecisionType" use="optional"/>
1608         </extension>
1609     </complexContent>
1610 </complexType>
1611 <element name="Actions" type="saml:ActionsType"/>
1612 <complexType name="ActionsType">
1613     <sequence>
1614         <element ref="saml:Action" maxOccurs="unbounded"/>
1615     </sequence>
1616     <attribute name="Namespace" type="anyURI" use="optional"/>

```



```

1617 </complexType>
1618 <element name="Action" type="string"/>
1619 <element name="Evidence" type="saml:AssertionSpecifierType"/>
1620 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1621 <complexType name="AttributeStatementType">
1622   <complexContent>
1623     <extension base="saml:SubjectStatementAbstractType">
1624       <sequence>
1625         <element ref="saml:Attribute" maxOccurs="unbounded"/>
1626       </sequence>
1627     </extension>
1628   </complexContent>
1629 </complexType>
1630 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
1631 <complexType name="AttributeDesignatorType">
1632   <attribute name="AttributeName" type="string"/>
1633   <attribute name="AttributeNamespace" type="anyURI"/>
1634 </complexType>
1635 <element name="Attribute" type="saml:AttributeType"/>
1636 <complexType name="AttributeType">
1637   <complexContent>
1638     <extension base="saml:AttributeDesignatorType">
1639       <sequence>
1640         <element ref="saml:AttributeValue"/>
1641       </sequence>
1642     </extension>
1643   </complexContent>
1644 </complexType>
1645 <element name="AttributeValue" type="saml:AttributeValueType"/>
1646 <complexType name="AttributeValueType">
1647   <sequence>
1648     <any namespace="##any" processContents="lax"
1649         minOccurs="0" maxOccurs="unbounded"/>
1650   </sequence>
1651 </complexType>
1652 </schema>
1653

```

8.2. Protocol Schema

1654

1655 Following is a complete listing of the SAML protocol schema [SAML-P-XSD].

```

1656 <?xml version="1.0" encoding="UTF-8"?>
1657 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1658 (VeriSign Inc.) -->
1659 <schema
1660   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
1661   sstc-schema-protocol-22.xsd"
1662   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1663   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1664   schema-assertion-22.xsd"
1665   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1666   schema-protocol-22.xsd"
1667   xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
1668   <import
1669     namespace="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1670     schema-assertion-22.xsd"
1671     schemaLocation="draft-sstc-schema-assertion-22.xsd"/>
1672   <import namespace="http://www.w3.org/2000/09/xmldsig#"
1673     schemaLocation="xmldsig-core-schema.xsd"/>
1674   <annotation>
1675     <documentation>draft-sstc-schema-protocol-22.xsd</documentation>
1676   </annotation>

```

```

1677 <simpleType name="StatusCodeType">
1678   <restriction base="string">
1679     <enumeration value="Success"/>
1680     <enumeration value="Failure"/>
1681     <enumeration value="Error"/>
1682     <enumeration value="Unknown"/>
1683   </restriction>
1684 </simpleType>
1685 <complexType name="RequestAbstractType" abstract="true">
1686   <sequence>
1687     <element ref="saml:RespondWith"
1688       minOccurs="0" maxOccurs="unbounded"/>
1689     <element ref="ds:Signature" minOccurs="0" maxOccurs="unbounded"/>
1690   </sequence>
1691   <attribute name="RequestID" type="saml:IDType" use="required"/>
1692   <attribute name="MajorVersion" type="integer" use="required"/>
1693   <attribute name="MinorVersion" type="integer" use="required"/>
1694 </complexType>
1695 <element name="RespondWith" type="anyURI"/>
1696 <element name="Request" type="saml:RequestType"/>
1697 <complexType name="RequestType">
1698   <complexContent>
1699     <extension base="saml:RequestAbstractType">
1700       <choice>
1701         <element ref="saml:Query"/>
1702         <element ref="saml:SubjectQuery"/>
1703         <element ref="saml:AuthenticationQuery"/>
1704         <element ref="saml:AttributeQuery"/>
1705         <element ref="saml:AuthorizationDecisionQuery"/>
1706         <element ref="saml:AssertionID" maxOccurs="unbounded"/>
1707         <element ref="saml:AssertionArtifact" maxOccurs="unbounded"/>
1708       </choice>
1709     </extension>
1710   </complexContent>
1711 </complexType>
1712 <element name="AssertionArtifact" type="string"/>
1713 <element name="Query" type="saml:QueryAbstractType"/>
1714 <complexType name="QueryAbstractType" abstract="true">
1715   <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>
1716 <complexType name="SubjectQueryAbstractType" abstract="true">
1717   <complexContent>
1718     <extension base="saml:QueryAbstractType">
1719       <sequence>
1720         <element ref="saml:Subject"/>
1721       </sequence>
1722     </extension>
1723   </complexContent>
1724 </complexType>
1725 <element name="AuthenticationQuery" type="saml:AuthenticationQueryType"/>
1726 <complexType name="AuthenticationQueryType">
1727   <complexContent>
1728     <extension base="saml:SubjectQueryAbstractType">
1729       <sequence>
1730         <element ref="saml:ConfirmationMethod" minOccurs="0"/>
1731       </sequence>
1732     </extension>
1733   </complexContent>
1734 </complexType>
1735 <element name="AttributeQuery" type="saml:AttributeQueryType"/>
1736 <complexType name="AttributeQueryType">
1737   <complexContent>
1738     <extension base="saml:SubjectQueryAbstractType">
1739       <sequence>

```

```

1740         <element ref="saml:AttributeDesignator"
1741             minOccurs="0" maxOccurs="unbounded" />
1742     </sequence>
1743 </extension>
1744 </complexContent>
1745 </complexType>
1746 <element name="AuthorizationDecisionQuery"
1747     type="samlp:AuthorizationDecisionQueryType" />
1748 <complexType name="AuthorizationDecisionQueryType">
1749     <complexContent>
1750         <extension base="samlp:SubjectQueryAbstractType">
1751             <sequence>
1752                 <element ref="saml:Actions" />
1753                 <element ref="saml:Evidence"
1754                     minOccurs="0" maxOccurs="unbounded" />
1755             </sequence>
1756             <attribute name="Resource" type="anyURI" />
1757         </extension>
1758     </complexContent>
1759 </complexType>
1760 <complexType name="ResponseAbstractType" abstract="true">
1761     <sequence>
1762         <element ref="ds:Signature" minOccurs="0" maxOccurs="unbounded" />
1763     </sequence>
1764     <attribute name="ResponseID" type="saml:IDType" use="required" />
1765     <attribute name="InResponseTo" type="saml:IDType" use="required" />
1766     <attribute name="MajorVersion" type="integer" use="required" />
1767     <attribute name="MinorVersion" type="integer" use="required" />
1768 </complexType>
1769 <element name="Response" type="samlp:ResponseType" />
1770 <element name="Response" type="samlp:ResponseType" />
1771 <complexType name="ResponseType">
1772     <complexContent>
1773         <extension base="samlp:ResponseAbstractType">
1774             <sequence>
1775                 <element ref="samlp:StatusReason"
1776                     minOccurs="0" maxOccurs="unbounded" />
1777                 <element ref="saml:Assertion"
1778                     minOccurs="0" maxOccurs="unbounded" />
1779             </sequence>
1780             <attribute name="StatusCode"
1781                 type="samlp:StatusCodeType" use="required" />
1782         </extension>
1783     </complexContent>
1784 </complexType>
1785 <element name="StatusReason" type="string" />
1786 </schema>
1787

```

9. References

1788

- 1789 [Kerberos] R. Needham et al., *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999, December 1978.
- 1790
- 1791
- 1792 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March 1984) Prentice Hall Computer Books;
- 1793
- 1794 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 1795
- 1796 [PKCS7] B. Kaliski., "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC 2315, March 1998.
- 1797
- 1798 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Service (V5)*. September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 1799
- 1800 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999. <http://www.ietf.org/rfc/rfc2246.txt>
- 1801
- 1802 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. June 1999. <http://www.ietf.org/rfc/rfc630.txt>
- 1803
- 1804 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999. <http://www.ietf.org/rfc/rfc2648.txt>
- 1805
- 1806 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 1807
- 1808 [RFC2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*, <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 1809
- 1810 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 1811
- 1812 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*, <http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf>, OASIS, December 2001.
- 1813
- 1814
- 1815
- 1816 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*, <http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf>, OASIS, December 2001.
- 1817
- 1818
- 1819
- 1820 [SAMLXSD] P. Hallam-Baker et al., *SAML protocol schema*, <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd>, OASIS, December 2001.
- 1821
- 1822
- 1823 [SAMLXSD] P. Hallam-Baker et al., *SAML assertion schema*, <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd>, OASIS, December 2001.
- 1824
- 1825
- 1826 [Schema1] H. S. Thompson et al., *XML Schema Part 1: Structures*, <http://www.w3.org/TR/xmlschema-1/>, World Wide Web Consortium Recommendation, May 2001.
- 1827
- 1828
- 1829 [Schema2] P. V. Biron et al., *XML Schema Part 2: Datatypes*, <http://www.w3.org/TR/xmlschema-2>, World Wide Web Consortium Recommendation, May 2001.
- 1830
- 1831
- 1832 [XMLEnc] *XML Encryption Specification*, In development.
- 1833 [XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*, <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.
- 1834

1835 **[XMLSig-XSD]** XML Signature Schema available from <http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd>.
1836
1837 **[XTAML]** P. Hallam-Baker, *XML Trust Axiom Markup Language 1.0*,
1838 <http://www.xmltrustcenter.org/>, VeriSign Inc. September 2001.

Appendix A. Notices

1839

1840 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1841 that might be claimed to pertain to the implementation or use of the technology described in this
1842 document or the extent to which any license under such rights might or might not be available;
1843 neither does it represent that it has made any effort to identify any such rights. Information on
1844 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1845 website. Copies of claims of rights made available for publication and any assurances of licenses to
1846 be made available, or the result of an attempt made to obtain a general license or permission for
1847 the use of such proprietary rights by implementors or users of this specification, can be obtained
1848 from the OASIS Executive Director.

1849 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1850 applications, or other proprietary rights which may cover technology that may be required to
1851 implement this specification. Please address the information to the OASIS Executive Director.

1852 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
1853 2001. All Rights Reserved.

1854 This document and translations of it may be copied and furnished to others, and derivative works
1855 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1856 published and distributed, in whole or in part, without restriction of any kind, provided that the above
1857 copyright notice and this paragraph are included on all such copies and derivative works. However,
1858 this document itself may not be modified in any way, such as by removing the copyright notice or
1859 references to OASIS, except as needed for the purpose of developing OASIS specifications, in
1860 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights
1861 document must be followed, or as required to translate it into languages other than English.

1862 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1863 successors or assigns.

1864 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1865 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1866 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
1867 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1868 PARTICULAR PURPOSE.