



Security Assertions Markup Language

Core Assertion Architecture

Phillip Hallam-Baker

VeriSign

Tim Moses

Entrust

Bob Morgan

University of Washington

Carlisle Adams

Entrust

Charles Knouse

Oblix

David Orchard

Jamcracker

Eve Maler

Sun

Irving Reid

Baltimore

Jeff Hodges

Oblix

Marlena Erdos

Tivoli

Nigel Edwards

Hewlett Packard

Prateek Mishra

Netegrity

Draft Version 0.9: June 20th 2001

18

19 Security Assertions Markup Language

20 Version 0.9

21 Table Of Contents

22	Table Of Contents	2
23	Executive Summary	3
24	1 XML Assertion and Request Syntax	3
25	1.1 Namespaces	3
26	1.2 SAML Assertion	3
27	1.2.1 Element <SAMLAssertionPackage>	5
28	1.3 Claims	7
29	1.3.1 Element <Claims>	7
30	1.3.2 Element <AssertionRef>	7
31	1.3.3 Element <Subject>	8
32	1.3.4 Element <Authenticator>	9
33	1.3.5 Element <DecisionClaim>	9
34	1.3.6 Element <AuthenticationClaim>	10
35	1.3.7 Element <AttributeClaim>	10
36	1.3.8 Element <ExtendedAttributeClaim>	10
37	1.3.9 Element <AuthorizationClaim>	11
38	1.4 Conditions	11
39	1.4.1 Element <Conditions>	11
40	1.4.2 Element <AudienceRestrictionCondition>	12
41	1.5 Advice	13
42	1.5.1 Element <Advice>	13
43	1.6 SAML Protocol	13
44	1.6.1 Abstract Type SAMLQueryType	14
45	1.6.2 Element <Respond>	14
46	1.6.3 Abstract Type SAMLSubjectQueryType	15
47	1.6.4 Element <SAMLAuthenticationQuery>	16
48	1.6.5 Element <SAMLAuthorizationQuery>	16
49	1.6.6 Element <SAMLAttributeQuery>	16
50	1.6.7 Element <SAMLQueryResponse>	16
51	1.7 Schema Extension	17
52	2 References	17

53

54 **Executive Summary**

55 This document contains two sections. Section 1 contains the text proposed by the Core
56 Assertions & Protocol group for the Core Assertions section of the SAML. Section 2
57 contains references to the material cited in the text.

58 **1 XML Assertion and Request Syntax**


59 **1.1 Namespaces**

60 In this document, certain namespace prefixes represent certain namespaces.

61 All SAML protocol elements are defined using XML schema **[XML-Schema1][XML-**
62 **Schema2]**. For clarity unqualified elements in schema definitions are in the XML schema
63 namespace:

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

65 References to Security Assertion Markup Language schema defined herein use the prefix
66 “s0” and are in the namespace:

```
67  xmlns:s0="http://www.oasis.org/tbs/1066-12-25/" [PHB1]
```

68 This namespace is also used for unqualified elements in message protocol examples.

69 The SAML schema specification uses some elements already defined in the XML
70 Signature namespace. The “XML Signature namespace” is represented by the prefix ds
71 and is declared as:

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

73 The “XML Signature schema” is defined in **[XML-SIG-XSD]** and the `<ds:KeyInfo>`
74 element (and all of its contents) are defined in **[XML-SIG]§4.4**.

```
75 <?xml version="1.0" encoding="UTF-8"?>  
76 <schema  
77     targetNamespace="http://www.oasis.org/tbs/1066-12-25/"  
78     xmlns="http://www.w3.org/2001/XMLSchema"  
79     xmlns:s0="http://www.oasis.org/tbs/1066-12-25/"  
80     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
81     elementFormDefault="unqualified">
```

82 **1.2 SAML Assertion**

83 SAML specifies several different types of assertion for different purposes, these are:

84 **Authentication Assertion**

85 An authentication assertion asserts that the issuer has authenticated the specified
86 subject.

87 **Attribute Assertion**

88 An attribute assertion asserts that the specified subject has the specified
89 attribute(s). Attributes may be specified by means of a URI or through an
90 extension schema that defines structured attributes.

91 **Decision Assertion**

92 A decision assertion reports the result of the specified authorization request.

93 **Authorization Assertion**

94 An authorization assertion asserts that a subject has been granted specific
95 permissions to access one or more resources.

96 The different types of SAML assertion are encoded in a common XML package, which at
97 a minimum consists of:

98 **Basic Information.**

99 Each assertion **MUST** specify a unique identifier that serves as a name for the
100 assertion. In addition an assertion **MAY** specify the date and time of issue and the
101 time interval for which the assertion is valid.

102 **Claims.**

103 The claims made by the assertion. This document describes the use of assertions
104 to make claims for Authorization and Key Delegation applications.

105 In addition an assertion **MAY** contain the following additional elements. An SAML
106 client is not required to support processing of any element contained in an additional
107 element **with the sole exception that an SAML client MUST reject any assertion**
108 **containing a Conditions element that is not supported.**

109 **Conditions.**

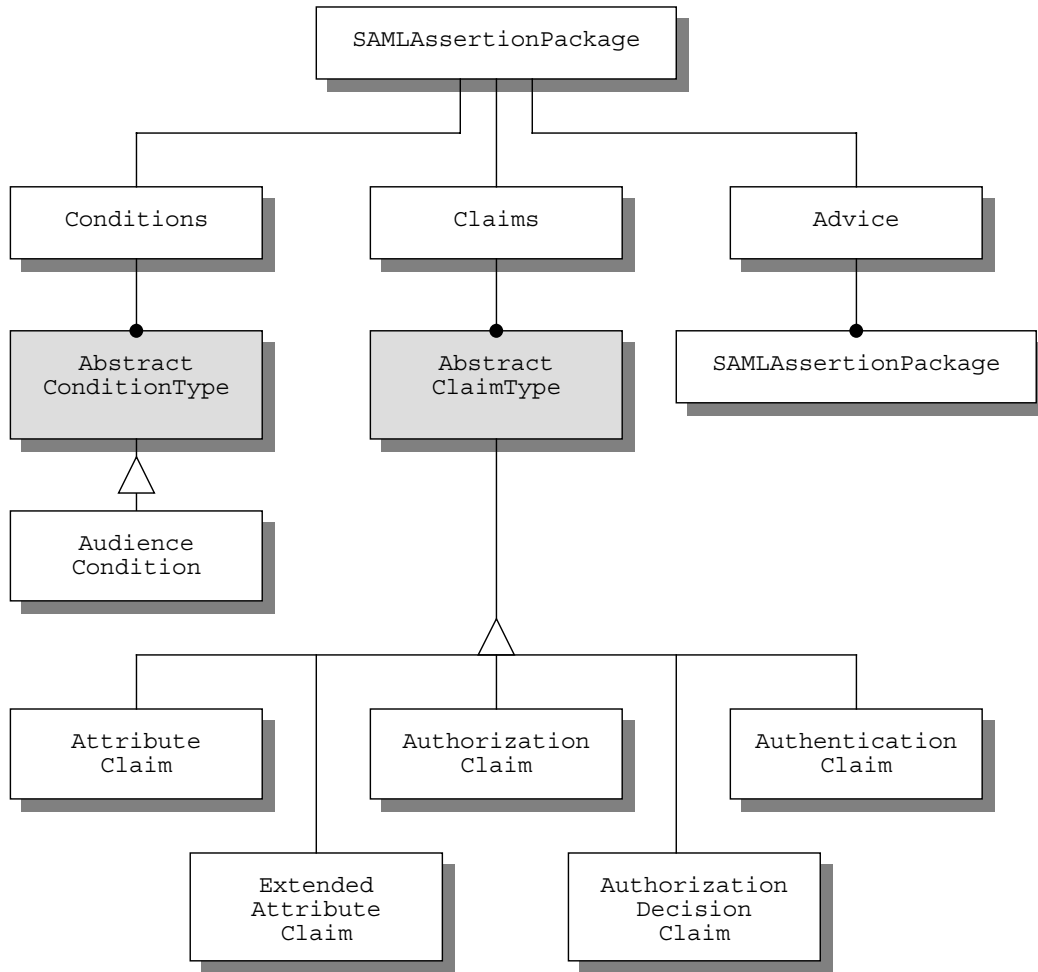
110 The assertion status **MAY** be subject to conditions. The status of the assertion
111 might be dependent on additional information from a validation service. The
112 assertion may be dependent on other assertions being valid. The assertion may
113 only be valid if the relying party is a member of a particular audience.

114 **Advice.**

115 Assertions **MAY** contain additional information as advice. The advice element
116 **MAY** be used to specify the assertions that were used to make a policy decision.

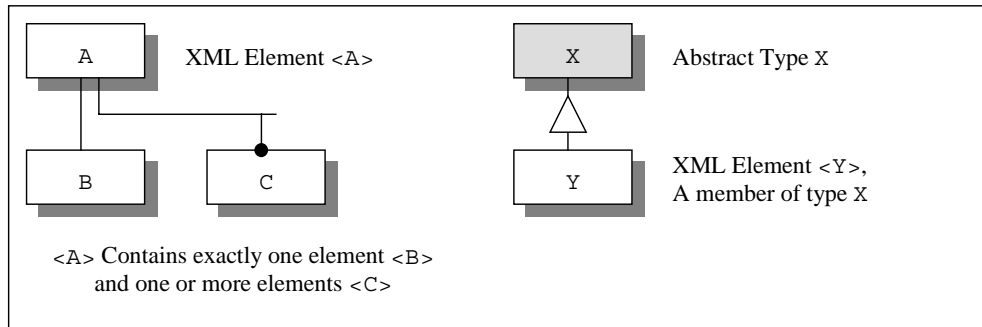
117 The SAML assertion package is designed to facilitate reuse in other specifications. For
118 this reason XML elements specific to the management of authentication and
119 authorization data are expressed as claims. Possible additional applications of the
120 assertion package format include management of embedded trust roots [**XTASS**] and
121 authorization policy information [**XACML**].

122 Figure 1 shows the class diagram of the SAMLAssertionPackage data structure. The
123 notation used in this diagram is described in Figure 2.



124
125

Figure 1: <SAMLAssertionPackage> Class Diagram



126
127

Figure 2: Key to Figure 1 and Figure 3

128 1.2.1 Element <SAMLAssertionPackage>

129 The <SAMLAssertionPackage> element is specified by the following schema:

```

130 <element name="SAMLAssertionPackage" type="S0:SAMLAssertionPackageType">
131
132 <complexType name="SAMLAssertionPackageType">
133 <!-- Basic Information -->

```

```
134     <attribute name="Version"                type="string"/>
135     <attribute name="AssertionID"           type="uriReference"/>
136     <attribute name="Issuer"                type="string"/>
137     <attribute name="IssueInstant"         type="timeInstant"/>
138     <attribute name="NotBefore"            type="timeInstant"/>
139     <attribute name="NotOnOrAfter"         type="timeInstant"/>
140
141     <element name="Claims"                  type="s0:Claims"      minOccurs="0"/>
142     <element name="Conditions"             type="s0:Conditions" minOccurs="0"/>
143     <element name="Advice"                  type="s0:Advice"      minOccurs="0"/>
144 </complexType>
```

145 Six basic information attributes are defined; a protocol version identifier, a unique
146 assertion identifier, an issuer identifier, the time instant of issue, the bounds of the
147 validity interval.

148 **1.2.1.1 Attribute Version**

149 Each assertion MUST specify the SAML version identifier. The identifier for this version
150 of SAML is the string "1.0".

151 **1.2.1.2 Attribute AssertionID**

152 Each assertion MUST specify exactly one unique assertion identifier. All identifiers are
153 encoded as a Uniform Resource Identifier (URI) and are specified in full (use of relative
154 identifiers is not permitted).

155 The URI is used as a *name* for the assertion and not as a *locator*. For the purposes of the
156 SAML protocol it is only necessary to ensure that no two assertions share the same
157 identifier. Provision of a service to resolve an identifier into an assertion is not a
158 requirement but applications MAY specify a URL as the assertion identifier that MAY
159 resolve to the assertion.

160 **1.2.1.3 Attribute Issuer**

161 The `Issuer` attribute specifies the issuer of the assertion by means of a URI.

162 **1.2.1.4 Attribute IssueInstant**

163 The `IssueInstant` attribute specifies the time instant of issue in Universal
164 Coordinated Time (UTC).

165 **1.2.1.5 Attribute NotBefore and NotOnOrAfter**

166 The `NotBefore` and `NotOnOrAfter` attributes specify limits on the validity of the
167 assertion.

168 The `NotBefore` attribute specifies the time instant at which the validity interval begins.
169 The `NotOnOrAfter` attribute specifies the time instant at which the validity interval
170 has ended

171 The `NotBefore` and `NotOnOrAfter` attributes are optional. If the value is either
172 omitted or equal to the start of the epoch it is unspecified. If the `NotBefore` attribute is
173 unspecified the assertion is valid at any time before the time instant specified by the
174 `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified the assertion
175 is valid from the time instant specified by the `NotBefore` attribute with no expiry. If
176 neither attribute is specified the assertion is valid at any time.

177 In accordance with the XML Schemas Specification, all time instances are interpreted in
178 Universal Coordinated Time unless they explicitly indicate a time zone.

179 Implementations **MUST NOT** generate time instances that specify leap seconds.

180 **1.3 Claims**

181 1.3.1 Element `<Claims>`

182 The `<Claims>` element contains one or more SAML assertion claims elements of type
183 `<AbstractClaimType>`.

184 In each case if more than one assertion claim element is specified the validity of each
185 claim is asserted jointly and severally, that is the semantics of a single assertion
186 containing two claims are identical to the semantics of two separate assertions each of
187 which contain one of the claims.

188 The following schema defines the `<Claims>` element:

```
189 <element name="Claims">  
190   <complexType>  
191     <sequence>  
192       <element name="AbstractClaim" type="s0:AbstractClaimType"  
193         minOccurs="0" maxOccurs="unbounded"/>  
194     </sequence>  
195   </complexType>  
196 </element>  
197  
198 <complexType name="AbstractClaimType" abstract="true">  
199   <sequence>  
200     <element name="AssertionRef" type="s0:AssertionRef"/>  
201     <!-- To add conditions on a per claim basis add :  
202     <element name="Conditions" type="s0:Conditions" minOccurs="0"/>  
203     -->  
204   </sequence>  
205 </complexType>
```

206 1.3.2 Element `<AssertionRef>`

207 The `<AssertionRef>` element specifies the assertion identifier of a prior assertion
208 that has been used to generate the assertion in which the `<AssertionRef>` element
209 occurs.

210 The primary purpose of `<AssertionRef>` elements is to permit auditing of SAML
211 applications. As such an `<AssertionRef>` element is advisory only and does not

212 mandate any specific action on the part of the application (such as tracking validity
213 dependencies).

214 The following elements may include `<AssertionRef>` elements:

215 **AbstractClaimType**

216 Advises that the specified claim was derived from the specified assertion.

217 **Subject**

218 Advises that the Subject definition of the claim was derived from the specified
219 assertion.

220 **Advice**

221 Advises that the referenced assertion was used to derive some unspecified portion
222 of the assertion.

223 The following schema defines the `<AssertionRef>` element:

```
224 <complexType name="URIReferenceType">  
225   <attribute name="id" type="uriReference"/>  
226 </complexType>  
227  
228 <element name="AssertionRef" type="s0:URIReferenceType"/>
```

229 **1.3.3 Element `<Subject>`**

230 The `<Subject>` element specifies the subject of the binding. In every case the subject
231 of a SAML assertion binding is a principal. A principal MAY be identified by name
232 and/or by reference to authentication credentials.

233 The following forms of subject name are supported:

Element	Description
<code><CommonName></code>	An unstructured text string, for example "Alice Aardvark".
<code><NameID></code>	A URI that specifies the principal by means of a machine-readable identifier.
<code><Authenticator></code>	Specifies credentials and an authentication protocol by which the subject may be identified.
<code><AssertionRef></code>	Specifies that the contents of the <code><Subject></code> element were derived from the specified assertion.
<code><AbstractSubject></code>	Extension schema...

234 In addition the principal MAY be specified by reference to authentication credentials by
235 means of the `<Authenticator>` element.

236 The following schema defines the `<Subject>` element:


```
237 <element name="Subject">
238   <complexType>
239     <sequence>
240       <element name="CommonName" type="string"/>
241       <element name="NameID" type="s0:URIReferenceType"/>
242       <element name="Authenticator" type="s0:Authenticator"/>
243       <element name="AssertionRef" type="s0:AssertionRef"/>
244       <element name="AbstractSubject"
245         type="s0:AbstractSubjectType"/>
246     </sequence>
247   </complexType>
248 </element>
249
250 <complexType name="AbstractSubjectType" abstract="true"/>
```

251 1.3.4 Element <Authenticator>

252 The <Authenticator> element specifies a means of identifying the subject of the
253 binding by means of their authentication credentials.

254 The authentication credentials MAY be specified either by means of the XML Digital
255 Signature <ds:KeyInfo> element or by means of the <Authdata> element.
256 Applications SHOULD make use of the <ds:KeyInfo> element for credentials that it
257 supports. Applications MAY use the <Authdata> element to specify other types of
258 authentication credentials, including passwords.

259 The <Authenticator> element MAY specify one or more <Protocol> elements.
260 If present the <Protocol> elements specify the authentication algorithms with which
261 the authentication credentials MAY be used to obtain an acceptable authentication.

262 The following schema defines the <Authenticator> element:

```
263 <element name="Authenticator">
264   <complexType>
265     <sequence>
266       <element name="Protocol" type="uriReference"
267         minOccurs="0" maxOccurs="unbounded"/>
268       <element name="Authdata" type="string"/>
269       <element name="KeyInfo" type="ds:KeyInfo"/>
270     </sequence>
271   </complexType>
272 </element>
```

273 1.3.5 Element <DecisionClaim>

274 The <DecisionClaim> element asserts that the access permissions specified in the
275 request identified by the corresponding RequestID were either permitted, denied or could
276 not be determined.

277 The following schema defines the <DecisionClaim> element:

```
278 <complexType name="DecisionClaim">
279   <complexContent>
280     <extension base="s0:AbstractClaimType">
281       <attribute name="Decision" type="s0:DecsionType"/>
282     </extension>
283   </complexContent>
284 </complexType>
```

```
285
286 <simpleType name=DecisionType>
287   <restriction base="string">
288     <enumeration value="Permit"/>
289     <enumeration value="Deny"/>
290     <enumeration value="Indeterminate"/>
291   </restriction>
292 </simpleType>
```

293 1.3.6 Element <AuthenticationClaim>

294 The <AuthenticationClaim> element asserts that the specified subject has been
295 authenticated.^[PHB2]



296 The following schema defines the <AuthenticationClaim> element:

```
297 <complexType name="AuthenticationClaim">
298   <complexContent>
299     <extension base="s0:AbstractClaimType">
300       <sequence>
301         <element name="Subject" type="s0:Subject"/>
302       </sequence>
303     </extension>
304   </complexContent>
305 </complexType>
```

306 1.3.7 Element <AttributeClaim>

307 The <AttributeClaim> element asserts that a specified subject has the specified
308 attribute(s) specified by a URI.

309 The following schema defines the <AttributeClaim> element:

```
310 <complexType name="AttributeClaim">
311   <complexContent>
312     <extension base="s0:AbstractClaimType">
313       <sequence>
314         <element name="Subject" type="s0:Subject"/>
315         <element name="AttributeID" type="s0:URIReferenceType"
316           minOccurs="0" maxOccurs="unbounded"/>
317       </sequence>
318     </extension>
319   </complexContent>
320 </complexType>
```

321 1.3.8 Element <ExtendedAttributeClaim>

322 The <ExtendedAttributeClaim> element asserts a relationship between the
323 specified subject and a collection of attributes specified by means of an extension
324 schema.

325 The following schema defines the <ExtendedAttributeClaim> element:

```
326 <complexType name="ExtendedAttributeClaim">
327   <complexContent>
328     <extension base="s0:AbstractClaimType">
329       <sequence>
330         <element name="Subject" type="s0:Subject"/>
331         <element name="Attribute" type="s0:AbstractAttributeType"
332           minOccurs="0" maxOccurs="unbounded"/>
332       </sequence>
332 </complexType>
```

```
333     </sequence>
334   </extension>
335 </complexContent>
336 </complexType>
337
338 <complexType name="AbstractAttributeType" abstract="true"/>
```

339 1.3.9 Element <AuthorizationClaim>

340 The <AuthorizationClaim> element asserts that the specified subject is authorized
341 to perform the specified operation(s) on the specified resource(s).

342 Defined permissions are Read, Write, Execute, Delete and Control. Additional
343 permissions may be specified by URI through an <ExtendedPermissions>
344 element.

345 The following schema defines the <AuthorizationClaim> element:

```
346 <complexType name="AuthorizationClaim">
347   <complexContent>
348     <extension base="s0:AbstractClaimType">
349       <sequence>
350         <element name="Subject" type="s0:Subject"/>
351         <element name="Authorization" type="s0:Authorization"
352           minOccurs="0" maxOccurs="unbounded"/>
353       </sequence>
354     </extension>
355   </complexContent>
356 </complexType>
357
358 <element name="Authorization">
359   <complexType>
360     <sequence>
361       <element name="Resource" type="uriReference"
362         minOccurs="0" maxOccurs="unbounded"/>
363       <element name="Permission" type="s0:PermissionType"
364         minOccurs="0" maxOccurs="unbounded"/>
365       <element name="ExtendedPermission" type="s0:URIReferenceType"
366         minOccurs="0" maxOccurs="unbounded"/>
367     </sequence>
368   </complexType>
369 </element>
370
371 <simpleType name="PermissionType">
372   <restriction base="string">
373     <enumeration value="Read"/>
374     <enumeration value="Write"/>
375     <enumeration value="Execute"/>
376     <enumeration value="Delete"/>
377     <enumeration value="Control"/>
378   </restriction>
379 </simpleType>
```

380 1.4 Conditions

381 1.4.1 Element <Conditions>

382 Assertion Conditions are contained in the <Conditions> element. SAML applications
383 MAY define additional elements using an extension schema. If an application encounters

384 an element contained within a <Conditions> element that is not understood the status
385 of the Condition MUST be considered Indeterminate.

386 The following schema defines the <Conditions> element:

```
387 <element name="Conditions">  
388   <complexType>  
389     <sequence>  
390       <element name="AbstractCondition"  
391         type="s0:AbstractConditionType"  
392         minOccurs="0" maxOccurs="unbounded"/>  
393     </sequence>  
394   </complexType>  
395 </element>  
396  
397 <complexType name="AbstractConditionType" abstract="true"/>
```

398 1.4.2 Element <AudienceRestrictionCondition>

399 Assertions MAY be addressed to a specific audience. Although a party that is outside the
400 audience specified is capable of drawing conclusions from an assertion, the issuer
401 explicitly makes no representation as to accuracy or trustworthiness to such a party.

- 402 • Require users of an assertion to agree to specific terms (rule book, liability caps,
403 relying party agreement)
- 404 • Prevent clients inadvertently relying on data that does not provide a sufficient
405 warranty for a particular purpose
- 406 • Enable sale of per-transaction insurance services.

407 An audience is identified by a URI that identifies to a document that describes the terms
408 and conditions of audience membership.

409 Each client is configured with a set of URIs that identify the audiences that the client is a
410 member of, for example:

411 <http://cp.verisign.test/cps-2000>
412 Client accepts the VeriSign Certification Practices Statement

413 http://rule.bizexchange.test/bizexchange_ruebook
414 Client accepts the provisions of the *bizexchange* rule book.

415 An assertion MAY specify a set of audiences to which the assertion is addressed. If the
416 set of audiences is the empty set there is no restriction and all audiences are addressed.
417 Otherwise the client is not entitled to rely on the assertion unless it is addressed to one or
418 more of the audiences that the client is a member of. For example:

419 <http://cp.verisign.test/cps-2000/part1>
420 Assertion is addressed to clients that accept the provisions of a specific part of the
421 VeriSign CPS.

422 In this case the client accepts a superset of the audiences to which the assertion is
423 addressed and may rely on the assertion.

424 The following schema defines the <AudienceRestrictionCondition> element:

```
425 <complexType name="AudienceRestrictionCondition">  
426   <complexContent>  
427     <extension base="s0:AbstractConditionType">  
428       <sequence>  
429         <element name="Audience" type="s0:URIReferenceType"  
430           minOccurs="0" maxOccurs="unbounded"/>  
431       </sequence>  
432     </extension>  
433   </complexContent>  
434 </complexType>
```

435 1.5 Advice

436 The Advice element is a general container for any additional information that does not
437 affect the semantics or validity of the assertion itself.

438 1.5.1 Element <Advice>

439 The <Advice> element permits evidence supporting the assertion claims to be cited,
440 either directly (through incorporating the claims) or indirectly (by reference to the
441 supporting assertions.

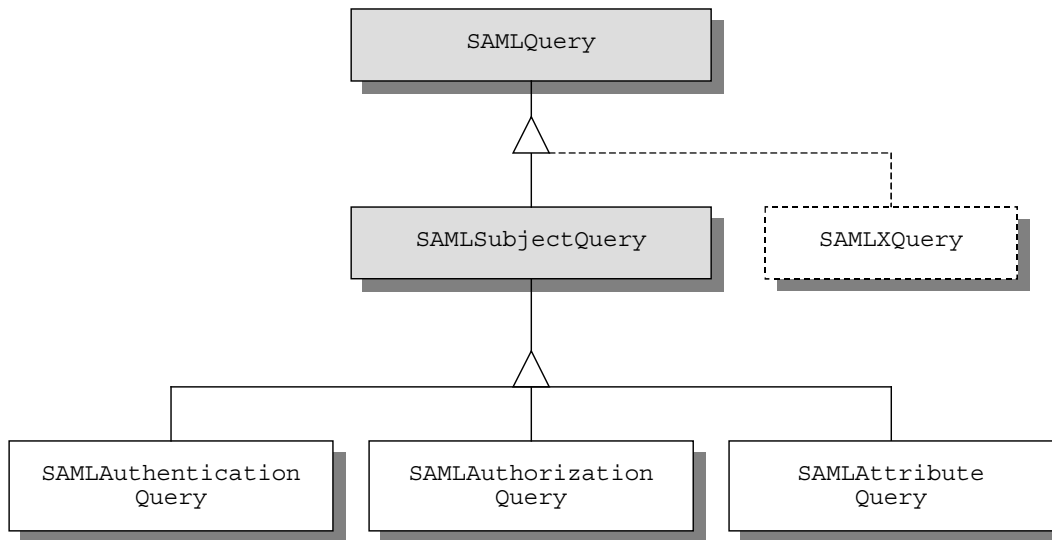
442 The following schema defines the <Advice> element:

```
443 <element name="Advice">  
444   <complexType>  
445     <sequence>  
446       <element name="Assertion" type="s0:Assertion"  
447         minOccurs="0" maxOccurs="unbounded"/>  
448       <element name="AssertionRef" type="s0:AssertionRef"  
449         minOccurs="0" maxOccurs="unbounded"/>  
450       <any namespace="##any" processContents="Skip">  
451     </sequence>  
452   </complexType>  
453 </element>
```

454 1.6 SAML Protocol

455 SAML Assertions may be generated and exchanged using a variety of protocols. The
456 bindings section of this document describes specific means of transporting SAML
457 assertions using existing widely deployed protocols.

458 SAML aware clients may in addition use the request protocol defined by the
459 <SAMLQuery> and <SAMLQueryResponse> elements described in this section.
460 Figure 3 shows the class diagram for the <SAMLQuery> element.




461

462

Figure 3: <SAMLQuery> Class Diagram

463 1.6.1 Abstract Type SAMLQueryType

464  All SAML Queries are extensions of the SAMLQueryType. Every query MUST
465 specify a RequestID attribute. The types of information that may be accepted in the
466 response are specified by means of the <Respond> element described in section 1.6.2.

467 The following schema defines the SAMLQueryType abstract type:

```
468 <complexType name="SAMLQueryType" abstract="true">
469   <sequence>
470     <attribute name="RequestID" type="uriReference"/>
471     <element name="Respond" type="s0:Respond"/>
472   </sequence>
473 </complexType>
```

474 1.6.1.1 Attribute <RequestID>

475 The RequestID attribute defines a unique identifier for the assertion request. If an
476 assertion query specifies a RequestID value the same value MUST be returned in the
477 response unless a Respond element of Static is specified.

478 1.6.2 Element <Respond>

479 The <Respond> element in the request specifies one or more strings included in the
480 request that specify data elements to be provided in the response.

481 The Service SHOULD return a requested data element if it is available. The Service
482 MAY return additional data elements that were not requested. In particular, the service
483 MAY return data elements specified in the request with the response.

484 Defined identifiers include:

Identifier	Description
Static	Specifies that the response may return any data element thus allowing the responder to return a static pre-signed assertion.
DecisionClaim	Specifies that the response may return an assertion that contains a <DecisionClaim> element
AttributeClaim	Specifies that the response may return an assertion that contains a <AttributeClaim> element
ExtendedAttributeClaim	Specifies that the response may return an assertion that contains a <ExtendedAttributeClaim> element
AuthorizationClaim	Specifies that the response may return an assertion that contains a <AuthorizationClaim> element
AuthenticationClaim	Specifies that the response may return an assertion that contains a <DecisionClaim> element
<i>XML Schema URI</i>	If a URI is specified the response may contain Claims, Conditions and Advice elements specified by the corresponding XML schema.


485 The following schema defines the <Respond> element:

```

486 <element name="Respond" >
487   <complexType>
488     <sequence>
489       <element name="Accept" type="string"
490         minOccurs="0" maxOccurs="unbounded"/>
491     </sequence>
492   </complexType>
493 </element>

```

494 1.6.3 Abstract Type SAMLSubjectQueryType

495  HB4] The SAMLSubjectQuery type extends the SAMLSubjectQuery type to
496 specify the a query with a specific subject as its principal.

497 The following schema defines the SAMLSubjectQuery abstract type:

```

498 <complexType name="SAMLSubjectQueryType" abstract="true">
499   <complexContent>
500     <extension base="s0:SAMLQueryType">
501     <sequence>

```

```
502         <element name="Subject"         type="s0:Subject"/>
503         <element name="Respond"        type="s0:Respond"/>
504     </sequence>
505 </extension>
506 </complexContent>
507 </complexType>
```

508 1.6.4 Element <SAMLAuthenticationQuery>

509  PHB5]

510 The following schema defines the SAMLAuthenticationQuery element:

```
511 <element name="SAMLAuthenticationQuery">
512     <complexContent>
513         <extension base="s0:SubjectQueryType">
514             </extension>
515         </complexContent>
516     </element>
```

517 1.6.5 Element <SAMLAuthorizationQuery>

518  PHB6]

519 The following schema defines the SAMLAuthorizationQuery element:

```
520 <element name="SAMLAuthorizationQuery">
521     <complexContent>
522         <extension base="s0:SubjectQueryType">
523             <sequence>
524                 <element name="Authorization" type="s0:Authorization"
525                     minOccurs="0" maxOccurs="unbounded"/>
526             </sequence>
527         </extension>
528     </complexContent>
529 </element>
```

530 1.6.6 Element <SAMLAttributeQuery>

531  PHB7]

532 The following schema defines the SAMLAttributeQuery element:

```
533 <element name="SAMLAttributeQuery">
534     <complexContent>
535         <extension base="s0:SubjectQueryType">
536             <sequence>
537                 <element name="AttributeID" type="s0:uriReferenceType"
538                     minOccurs="0" maxOccurs="unbounded"/>
539                 <element name="Authorization" type="s0:Authorization"
540                     minOccurs="0" maxOccurs="unbounded"/>
541             </sequence>
542         </extension>
543     </complexContent>
544 </element>
```

545 1.6.7 Element <SAMLQueryResponse>

546 The response to a <SAMLQuery> is a <SAMLQueryResponse> element. This returns
547 the Request ID specified in the response and a <SAMLAssertionPackage>

548 element. The information returned in the response is controlled by the <Respond>
549 element of the request.

550 The <SAMLQueryResponse> element is defined by the following schema:

```
551 <element name="SAMLQueryResponse">  
552   <complexType>  
553     <sequence>  
554       <!-- Basic Information -->  
555       <attributename="RequestID" type="s0:uriReference"/>  
556       <element name="SAMLAssertionPackage"  
557         type="s0:SAMLAssertionPackageType"/>  
558     </sequence>  
559   </complexType>  
560 </element>  
561 </schema>
```

563 1.7 Schema Extension

564 The SAML schema is designed to support extensibility by means of XML abstract types.
565 Extension schemas should specify the purpose of extension elements by defining them as
566 extensions of the appropriate abstract types.

567 The following abstract types are defined in the schema:

Abstract Type	Purpose
AbstractClaimType	Specify a new claim element.
AbstractSubjectType	Specify a new element for identifying the subject of a claim.
AbstractAttributeType	Specify structured attribute data.
AbstractConditionType	Specify a new condition element.

568 In addition the <Advice> element permits arbitrary elements to be included without
569 type restriction.

570 2 References

- 571 [Kerberos] TBS
- 572 [SAML-USE] TBS
- 573 [PKCS1] Kaliski, B., *PKCS #1: RSA Encryption Version 2.0*, RSA
574 Laboratories, also IETF RFC 2437, October 1998.
- 575 [RFC-2104] Krawczyk, H., Bellare, M. and R. Canetti, *HMAC: Keyed Hashing*
576 *for Message Authentication*, IETF RFC 2104, February 1997.
- 577 [SOAP] D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn,
578 H. Frystyk Nielsen, S Thatte, D. Winer. *Simple Object Access*

- 579 *Protocol (SOAP) 1.1*, W3C Note 08 May 2000,
580 <http://www.w3.org/TR/SOAP>
- 581 **[WSSL]** E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web*
582 *Services Description Language (WSDL) 1.0* September 25, 2000,
583 <http://msdn.microsoft.com/xml/general/wsdl.asp>
- 584 **[XACML]** *TBS*
- 585 **[XTASS]** P. Hallam-Baker, *XML Trust Axiom Service Specification 1.0*,
586 VeriSign Inc. January 2001. <http://www.xmltrustcenter.org/>
- 587 **[XML-SIG]** D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon.
588 *XML-Signature Syntax and Processing*, World Wide Web
589 Consortium. <http://www.w3.org/TR/xmlsig-core/>
- 590 **[XML-SIG-XSD]** XML Signature Schema available from
591 [http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)
592 [core-schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).
- 593 **[XML-Enc]** *XML Encryption Specification*, In development.
- 594 **[XML-Schema1]** H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn. *XML*
595 *Schema Part 1: Structures*, W3C Working Draft 22 September
596 2000, <http://www.w3.org/TR/2000/WD-xmlschema-1-20000922/>,
597 latest draft at <http://www.w3.org/TR/xmlschema-1/>
- 598 **[XML-Schema2]** P. V. Biron, A. Malhotra, *XML Schema Part 2: Datatypes*; W3C
599 Working Draft 22 September 2000,
600 <http://www.w3.org/TR/2000/WD-xmlschema-2-20000922/>, latest
601 draft at <http://www.w3.org/TR/xmlschema-2/>

Page: 3

[PHB1] We have to align with the OASIS convention here.

Page: 10

[PHB2] I really think that this assertion is bogus as specified. An authentication assertion must have an object to make sense.

Page: 14

[PHB3] asking for it to be created.

Page: 15

[PHB4] asking for it to be created.

Page: 16

[PHB5] asking for it to be created.

Page: 16

[PHB6] asking for it to be created.

Page: 16

[PHB7] asking for it to be created.