

## 2 **Security Assertions Markup Language**

### 3 *Core Assertion Architecture*

4 *Phillip Hallam-Baker*

*VeriSign*

5 *Tim Moses*

*Entrust*

6 *Bob Morgan*

*University of Washington*

7 *Carlisle Adams*

*Entrust*

8 *Charles Knouse*

*Oblix*

9 *David Orchard*

*Jamcracker*

10 *Eve Maler*

*Sun*

11 *Irving Reid*

*Baltimore*

12 *Jeff Hodges*

*Oblix*

13 *Marlena Erdos*

*Tivoli*

14 *Nigel Edwards*

*Hewlett Packard*

15 *Prateek Mishra*

*Netegrity*

16 *Chris McLaren*

*Netegrity*

17 *Draft Version 0.15: August 21st 2001*

19

# 20 Security Assertions Markup Language

21 Version 0.15

## 22 Table Of Contents

23	Table Of Contents	2
24	<b>1 XML Assertion Syntax</b>	<b>4</b>
25	1.1 Namespaces	5
26	1.1.1 Basic Types	5
27	1.2 SAML Assertion	7
28	1.2.1 Abstract Element <Assertion>	7
29	1.2.2 Element <AssertionSpecifier>	8
30	1.3 Subject Assertion	8
31	1.3.1 Abstract Type SubjectAssertionAbstractType	8
32	1.3.2 Element <Subject>	8
33	1.4 Authentication Assertion	10
34	1.4.1 Assertion Type AuthenticationAssertionType	10
35	1.5 Authorization Decision Assertion	11
36	1.5.1 Assertion Type AuthorizationDecisionAssertionType	11
37	1.6 Attribute Assertion	12
38	1.6.1 Assertion Type AttributeAssertionType	12
39	1.7 Conditions	14
40	1.7.1 Element <Conditions>	14
41	1.7.2 Condition Type AudienceRestrictionConditionType	15
42	1.8 Advice	15
43	1.8.1 Element <Advice>	16
44	1.9 Schema Extension	16
45	<b>2 SAML Protocol</b>	<b>17</b>
46	2.1 Namespaces	17
47	2.1.1 Basic Types	17
48	2.2 Request	19
49	2.2.1 Abstract Type RequestAbstractType	19
50	2.2.2 Element <Request>	19
51	2.2.3 Abstract Type QueryAbstractType	19
52	2.2.4 Abstract Type SubjectQueryAbstractType	20
53	2.3 Authentication Query	20
54	2.3.1 Subject Query Type AuthenticationQueryType	20
55	2.4 Attribute Query	21
56	2.4.1 Subject Query Type AttributeQueryType	21
57	2.5 Authorization Query	21
58	2.5.1 Subject Query Type AuthorizationQueryType	21
59	2.6 Response	22

60	2.6.1	Abstract Type ResponseAbstractType	22
61	2.6.2	Element <Response>	23
62	2.7	Schema Extension	23
63	<b>3</b>	<b>References</b>	<b>24</b>
64	<b>4</b>	<b>Identifiers</b>	<b>25</b>
65	4.1	Authentication Protocol Identifiers	25
66	4.1.1	SAML Artifact	25
67	4.1.2	Assertion Bearer	25
68	4.1.3	User Name and Password (Pass-through)	25
69	4.1.4	User Name and Password (One-Way-Function SHA-1)	25
70	4.1.5	Kerberos	25
71	4.1.6	SSL/TLS Certificate Based Client Authentication	25
72	4.1.7	Object Authenticator (SHA-1)	25
73	4.2	Action Identifiers	25
74	4.2.1	Read/Write/Execute/Delete/Control	25
75	4.2.2	Read/Write/Execute/Delete/Control with Negation	25
76	4.2.3	Get/Head/Put/Post	25
77	4.2.4	UNIX file Permissions	25
78	<b>5</b>	<b>Appendix</b>	<b>26</b>
79	5.1	Assertion Schema	26
80	5.2	Protocol Schema	26
81			

## 82 **1 XML Assertion Syntax**

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

### 84 **Authentication Assertion**

85 An authentication assertion is an assertion by the issuer that the subject was  
86 authenticated by a particular means at a particular time.

### 87 **Authorization Decision Assertion**

88 An authorization decision assertion is an assertion by the issuer that the request  
89 for access by the specified subject to the specified object has resulted in the  
90 specified decision on the basis of some optionally specified evidence.

### 91 **Attribute Assertion**

92 An attribute assertion asserts that the specified subject is associated with the  
93 specified attribute(s).

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

### 96 **Basic Information.**

97 Each assertion **MUST** specify the version of the SAML assertion syntax, a unique  
98 identifier that serves as a name for the assertion, a unique identifier for the issuer  
99 and the time instant of issue.

### 100 **The Asserted Statement**

101 The statement that is asserted by the issuer of the assertion.

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

### 106 **Conditions.**

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

### 111 **Advice.**

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

114 The SAML assertion package is designed to facilitate reuse in other specifications. For  
115 this reason XML elements specific to the management of authentication and  
116 authorization data are expressed as claims. Possible additional applications of the

117 assertion package format include management of embedded trust roots [XTASS] and  
118 authorization policy information [XACML].

119 [Class diagram to be inserted][PHB1]

## 120 1.1 Namespaces

121 In this document, certain namespace prefixes represent specific XML namespaces.

122 All SAML protocol elements are defined using XML schema [XML-Schema1][XML-  
123 Schema2]. For clarity unqualified elements in schema definitions are in the XML schema  
124 namespace:

```
125 xmlns="http://www.w3.org/2001/XMLSchema"  
126 xmlns:xsd="http://www.w3.org/2001/XMLSchema"[PHB2]
```

127 References to Security Assertion Markup Language schema defined herein use the prefix  
128 saml : and are in the namespace:

```
129 xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-  
130 sstc-schema-assertion-15.xsd"
```

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

132 The XML schema specification uses some elements already defined in the XML  
133 Signature namespace. The “XML Signature namespace” is represented by the prefix ds  
134 and is declared as:

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

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

138 The following schema defines the XML namespaces for the assertion schema:

```
139 <?xml version="1.0" encoding="UTF-8"?>  
140 <schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/"  
141 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
142 xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"  
143 xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-  
144 sstc-schema-assertion-15.xsd"  
145 xmlns="http://www.w3.org/2000/10/XMLSchema"  
146 elementFormDefault="unqualified">  
147 <import namespace="http://www.w3.org/2000/09/xmldsig#"  
148 schemaLocation="xmldsig-core-schema.xsd"/>  
149 <annotation>  
150 <documentation>draft-schema-consensus-10.xsd</documentation>  
151 </annotation>
```

### 152 1.1.1 Basic Types

153 The types defined in this section define XML types that are considered part of the schema  
154 as a whole rather than a component of a particular element. This allows for greater  
155 consistency and avoids the need for extension schemas to redefine the same types.

### 156 **1.1.1.1 Simple Type IDType**

157 The IDType type is used for any data element that is an identifier for a specific data  
158 object that is opaque to the SAML application. In the SAML specification the IDType  
159 type is used declare and reference identifiers to assertions, requests and responses.

160 IDType identifiers MUST satisfy the following properties:

- 161 • Any party that assigns an IDType identifier to a data object MUST ensure that  
162 there is negligible probability that that party or any other party will assign the  
163 same identifier to a different data object.
- 164 • Where a data object specifies an IDType identifier that is to be an identifier for  
165 that object there MUST be exactly one such declaration.

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

170 An IDType identifier MAY or MAY NOT be resolvable. In the case that the identifier is  
171 resolvable (e.g. the identifier is a URL) the identifier MAY or MAY NOT resolve to the  
172 data object identified.

173 The <AssertionID> element is used to specify an identifier that references a SAML  
174 assertion.

175 The following schema specifies the <AssertionID> element and IDType type:

```
176 <element name="AssertionID" type="saml:IDType"/>  
177 <simpleType name="IDType">  
178 <restriction base="string"/>  
179 </simpleType>
```

### 180 **1.1.1.2 Simple Type DecisionType**

181 The DecisionType type reports the status of an authorization decision with respect to  
182 a specific resource.

#### 183 **Permit**

184 The specified action is permitted.

#### 185 **Deny**

186 The specified action is denied.

#### 187 **Indeterminate**

188 No statement is made as to whether the specified action is permitted or denied.

189 The following schema specifies the DecisionType type:

```
190 <simpleType name="DecisionType">  
191 <restriction base="string">
```

```
192     <enumeration value="Permit"/>
193     <enumeration value="Deny"/>
194     <enumeration value="Indeterminate"/>
195   </restriction>
196 </simpleType>
```

## 197 **1.2 SAML Assertion**

198 A SAML Assertion is specified by a single XML element whose type is derived from the  
199 abstract XML type `AssertionAbstractType`.<sup>[PHB3]</sup> The abstract  
200 `AssertionAbstractType` specifies the basic information that is common to all  
201 SAML assertions. Instances of SAML assertions have concrete types that are extensions  
202 of the base `AssertionAbstractType`.

### 203 **1.2.1 Abstract Element <Assertion>**

204 The element `<Assertion>` of abstract type `AssertionAbstractType` is used to  
205 specify a SAML assertion.

206 Following [XML-Schema] each instance of an `<Assertion>` element MUST specify  
207 the specific concrete type using the `xsiType` attribute:

```
208   <Assertion xsiType=AttributeAssertionType>
```

209 The following schema specifies the `<Assertion>` element and  
210 `AssertionAbstractType` abstract type:

```
211   <element name="Assertion" type="saml:AssertionAbstractType"/>
212   <complexType name="AssertionAbstractType" abstract="true">
213     <sequence>
214       <element name="Conditions" type="saml:ConditionsType" minOccurs="0"/>
215       <element name="Advice" type="saml:AdviceType" minOccurs="0"/>
216     </sequence>
217     <attribute name="Version" type="string" use="required"/>
218     <attribute name="AssertionID" type="saml:IDType" use="required"/>
219     <attribute name="Issuer" type="string" use="required"/>
220     <attribute name="IssueInstant" type="timeInstant" use="required"/>
221   </complexType>
222
```

223 The following basic information attributes are defined; a protocol version identifier, a  
224 unique assertion identifier, an issuer identifier and the time instant of issue.

#### 225 **1.2.1.1 Attribute Version**

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

#### 228 **1.2.1.2 Attribute AssertionID**

229 The `AssertionID` attribute specifies the assertion identifier.


230 **1.2.1.3 Attribute Issuer**

231 The Issuer attribute specifies the issuer of the assertion by means of a string<sup>[PHB4]</sup>.

232 **1.2.1.4 Attribute IssueInstant**

233 The IssueInstant attribute specifies the time instant of issue in Universal  
234 Coordinated Time (UTC).

235 **1.2.2 Element <AssertionSpecifier>**

236  <AssertionSpecifier> element specifies an assertion either by reference or  
237 by value. An assertion is specified by reference to the value of its AssertionID attribute.  
238 An assertion is specified by value by including the entire assertion.

239 The following schema specifies the <AssertionSpecifier> element:

```
240 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
241 <complexType name="AssertionSpecifierType">
242   <choice>
243     <element ref="saml:AssertionID" />
244     <element ref="saml:Assertion"/>
245   </choice>
246 </complexType>
```

247 **1.3 Subject Assertion**

248 A Subject Assertion is an assertion by the issuer that concerns a SAML subject specified  
249 by a <Subject> element.

250 **1.3.1 Abstract Type SubjectAssertionAbstractType**

251 The SubjectAssertionAbstractType type is an abstract type that extends the  
252 AssertionAbstractType to include a <Subject> element.

253 The following schema defines the SubjectAssertionAbstractType abstract  
254 type:

```
255 <complexType name="SubjectAssertionAbstractType" abstract="true">
256   <complexContent>
257     <extension base="saml:AssertionAbstractType">
258       <sequence>
259         <element ref="saml:Subject"/>
260       </sequence>
261     </extension>
262   </complexContent>
263 </complexType>
```

264 **1.3.2 Element <Subject>**

265 The <Subject> element specifies a party by any of the following means:

- 266
- A name.



- 267       • By information that allows the party to be authenticated.
- 268       • By reference to another assertion or by containment of another assertion.

269 If a <Subject> element contains more than one subject specification the issuer is  
270 asserting that all the subject specifications present specify the same subject. For example  
271 if both a <NameIdentifier> and a <SubjectConfirmation> element are  
272 present the issuer is asserting that the authentication data authenticates the party with the  
273 specified name.

274 The following schema defines the <Subject> element:

```
275 <element name="Subject" type="saml:SubjectType"/>
276 <complexType name="SubjectType">
277   <choice maxOccurs="unbounded">
278     <element ref="saml:NameIdentifier"
279       minOccurs="0" maxOccurs="unbounded"/>
280     <element ref="saml:SubjectConfirmation"
281       minOccurs="0" maxOccurs="unbounded"/>
282     <element ref="saml:AssertionSpecifier"
283       minOccurs="0" maxOccurs="unbounded"/>
284   </choice>
285 </complexType>
```

### 286 1.3.2.1 Element <SubjectConfirmation>

287 The <SubjectConfirmation> element specifies a subject by specifying data that  
288 authenticates the subject.

289       **<AuthenticationMethod>** [Any number]  
290       Each <AuthenticationMethod> element specifies a URI that identify a  
291       protocol that may be used to authenticate the subject.

292       **<SubjectConfirmationData>** [Optional]  
293       Each <SubjectConfirmationData> element specifies additional  
294       authentication information used by a specific authentication protocol.

295       **<ds:KeyInfo>** [Optional]  
296       An XML Signature <ds:KeyInfo> element that specifies a cryptographic key  
297       held by the subject.

298 URIs identifying common authentication protocols are specified in Section 4 .

299 The following schema defines the <SubjectConfirmation> element:

```
300 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
301 <complexType name="SubjectConfirmationType">
302   <sequence>
303     <element ref="saml:AuthenticationMethod" maxOccurs="unbounded"/>
304     <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
305     <element ref="ds:KeyInfo" minOccurs="0"/>
306   </sequence>
307 </complexType>
```

308 **1.3.2.2 Element <AuthenticationMethod>**

309 The <AuthenticationMethod> element specifies the type of Authentication that  
310 took place.

311 The following schema defines the <AuthenticationMethod> element:

```
312 <element name="AuthenticationMethod" type="uriReference"/>
```

313 **1.3.2.3 Element <NameIdentifier>**

314 The NameIdentifier type specifies a subject by a combination of a name and a security  
315 domain.



316 The interpretation of the security domain and the name are left to individual  
317 implementations.[PHB5]

318 The following schema defines the <NameIdentifier> element:

```
319 <element name="NameIdentifier" type="saml:NameIdentifierType"/>  
320 <complexType name="NameIdentifierType">  
321 <sequence>  
322 <element name="SecurityDomain" type="string"/>  
323 <element name="Name" type="string"/>  
324 </sequence>  
325 </complexType>
```

326 **1.4 Authentication Assertion**

327 An authentication assertion is an assertion by the issuer that the subject was authenticated  
328 by a particular means at a particular time.

329 **1.4.1 Assertion Type AuthenticationAssertionType**

330 The AuthenticationAssertionType extends the  
331 SubjectAssertionAbstractType with the addition of the following elements:

332 **<AuthenticationMethod>** [Required]

333 The <AuthenticationMethod> element specifies the type of Authentication  
334 that took place.

335 **<AuthenticationInstant>** [Required]

336 The <AuthenticationInstant> element specifies the time at which the  
337 authentication took place.

338 **<AuthenticationLocale>** [Optional]

339 The <AuthenticationLocale> element specifies the DNS domain name  
340 and IP address for the system entity that performed the authentication.

341 The following schema defines the <AuthenticationAssertion> assertion type:

```
342 <complexType name="AuthenticationAssertionType">  
343 <complexContent>
```

```
344     <extension base="saml:SubjectAssertionAbstractType">
345         <sequence>
346             <element ref="saml:AuthenticationMethod"/>
347             <element name="AuthenticationInstant" type="timeInstant"/>
348             <element name="AuthenticationLocale"
349                 type="saml:AuthenticationLocaleType" minOccurs="0"/>
350         </sequence>
351     </extension>
352 </complexContent>
353 </complexType>
```

#### 354 1.4.1.1 Type AuthenticationLocale

355 The <AuthenticationLocale> element specifies the DNS domain name and IP  
356 address for the system entity that performed the authentication.

357 *Note: This element is entirely advisory, since both these fields are quite easily “spoofed”*  
358 *but current practice appears to require its inclusion.*

359 The following schema defines the <AuthenticationLocale> type:

```
360 <complexType name="AuthenticationLocaleType">
361     <sequence>
362         <element name="IP" type="string" minOccurs="0"/>
363         <element name="DNS_Domain" type="string" minOccurs="0"/>
364     </sequence>
365 </complexType>
```

### 366 1.5 Authorization Decision Assertion

367 An authorization decision assertion is an assertion by the issuer that the request for access  
368 by the specified subject to the specified object has resulted in the specified decision on  
369 the basis of some optionally specified evidence.

#### 370 1.5.1 Assertion Type AuthorizationDecisionAssertionType

371 The AuthorizationDecisionAssertionType extends the  
372 SubjectAssertionAbstractType with the addition of the following elements:

373 **<Resource>** [Required]

374 The <Resource> element specifies the resource by means of a URI.

375 **<Namespace>** [Optional]

376 The <Namespace> element specifies the namespace in which the specified  
377 action elements are to be interpreted.

378 **<Action>** [One or more]

379 The <Action> element specifies the set of actions on the specified resource.

380 **<Decision>** [Required]

381 The <Decision> element specifies the decision with respect to the specified  
382 object.

383       **<Evidence>** [Optional]

384           The <Evidence> element specifies a set of assertions that the issuer relied upon  
385           in making the decision.

386   If the <Namespace> element is not specified the namespace specified in section 4.2.1 is  
387   specified by default.

388   The following schema defines the <AuthorizationDecisionAssertionType >  
389   type:

```
390       <complexType name="AuthorizationDecisionAssertionType">  
391           <complexContent>  
392               <extension base="saml:SubjectAssertionAbstractType">  
393                   <sequence>  
394                       <element name="Resource" type="xsd:uriReference"/>  
395                       <element name="Namespace" type="uriReference" minOccurs="0"/>  
396                       <element name="Action" type="string" maxOccurs="unbounded"/>  
397                       <element name="Decision" type="saml:DecisionType"/>  
398                       <element name="saml:Evidence"  
399                           minOccurs="0" maxOccurs="unbounded"/>  
400                   </sequence>  
401               </extension>  
402           </complexContent>  
403       </complexType>
```

#### 404   **1.5.1.1   Element <Evidence>**

405   The <Evidence> element specifies a set of assertions that the issuer relied upon in  
406   issuing the assertion.

407   The statement of an assertion as evidence MAY affect the reliance agreement between  
408   the client and service. For example in the case that the client presented an assertion to the  
409   service in a request the service MAY use that assertion as evidence in making its  
410   response without endorsing the assertion as valid either to the client or any third party.

411   The following schema defines the <Evidence> element:

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

### 413   **1.6   Attribute Assertion**

414   An attribute assertion asserts that the specified subject is associated with the specified  
415   attribute(s)

#### 416   **1.6.1   Assertion Type AttributeAssertionType**

417   The AttributeAssertionType extends the  
418   SubjectAssertionAbstractType with the addition of the following element:

419       **<Attribute>** [One or More]

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

421   The following schema defines the AttributeAssertionType assertion type:

```
422       <complexType name="AttributeAssertionType">
```

```
423     <complexContent>
424         <extension base="saml:SubjectAssertionAbstractType">
425             <sequence>
426                 <element ref="saml:Attribute" maxOccurs="unbounded"/>
427             </sequence>
428         </extension>
429     </complexContent>
430 </complexType>
```

### 431 1.6.1.1 Element <Attribute> and Element <AttributeNamespaceName>

432 The <Attribute> element specifies an attribute of the assertion subject. An attribute  
433 is identified by a name that is interpreted within a particular namespace. The  
434 <Attribute> element contains the following elements:

435 **<AttributeNamespace>** [Optional]

436 The <AttributeNamespace> element specifies the namespace in which the  
437 <AttributeName> elements are interpreted.

438 **<AttributeName>** [Required]

439 The <AttributeName> element specifies the name of the attribute.

440 **<AttributeValue>** [One or more]

441 Each <AttributeValue> element specifies a value of the attribute.

442 If no <AttributeNamespace> element is specified the interpretation of  
443 <AttributeName> elements is left to the implementation.

444 The <AttributeNamespaceName> element specifies the <AttributeName> and  
445 <AttributeNamespace> of an attribute but does not specify an  
446 <AttributeValue>.

447 The following schema defines the <Attribute> element:

```
448     <element name="AttributeName" type="string"/>
449     <element name="AttributeNamespace" type="uriReference"/>
450     <element name="Attribute" type="saml:AttributeType"/>
451     <complexType name="AttributeType">
452         <sequence>
453             <element ref="saml:AttributeName"/>
454             <element ref="saml:AttributeNamespace" minOccurs="0"/>
455             <element name="AttributeValue" type="saml:AttributeValueType"
456 maxOccurs="unbounded"/>
457         </sequence>
458     </complexType>
459     <element name="AttributeNamespaceName"
460         type="saml:AttributeNamespaceNameType"/>
461     <complexType name="AttributeNamespaceNameType">
462         <sequence>
463             <element ref="saml:AttributeName"/>
464             <element ref="saml:AttributeNamespace" minOccurs="0"/>
465         </sequence>
466     </complexType>
```

### 467 **1.6.1.2 Type AttributeValueType**

468 An <AttributeValue> element is of type AttributeValueType. The  
469 AttributeValueType allows the inclusion of any element in any namespace.[PHB6]

470 The following schema defines the AttributeValueType type:

```
471 <complexType name="AttributeValueType">  
472 <sequence>  
473 <any namespace="##any" processContents="lax"  
474 minOccurs="0" maxOccurs="unbounded"/>  
475 </sequence>  
476 </complexType>
```

## 477 **1.7 Conditions**

478 The validity of an assertion MAY be subject to a set of conditions. Each condition  
479 evaluates to a value that is Valid, Invalid or Indeterminate. The validity status  
480 of an assertion is the conjunction of the validity of each of the assertion conditions as  
481 follows:

482 If any condition evaluates to Invalid.  
483 The assertion status is Invalid

484 If no condition evaluates to Invalid and one or more conditions evaluate to  
485 Indeterminate.  
486 The assertion status is Indeterminate

487 If no conditions are specified or all the specified assertions evaluate to Valid.  
488 The assertion status is Valid

### 489 **1.7.1 Element <Conditions>**

490 Assertion Conditions are contained in the <Conditions> element. SAML applications  
491 MAY define additional elements using an extension schema. If an application encounters  
492 an element contained within a <Conditions> element that is not understood the status  
493 of the Condition MUST be considered Indeterminate.

494 The following schema defines the <Conditions> element:

```
495 <complexType name="ConditionsType">  
496 <sequence>  
497 <element name="Condition" type="saml:AbstractConditionType"  
498 minOccurs="0" maxOccurs="unbounded"/>  
499 </sequence>  
500 <attribute name="NotBefore" type="timeInstant" use="optional"/>  
501 <attribute name="NotOnOrAfter" type="timeInstant" use="optional"/>  
502 </complexType>  
503  
504 <complexType name="AbstractConditionType" abstract="true"/>
```

### 505 **1.7.1.1 Attributes NotBefore and NotOnOrAfter**

506 The NotBefore and NotOnOrAfter attributes specify limits on the validity of the  
507 assertion.

508 The NotBefore attribute specifies the time instant at which the validity interval begins.  
509 The NotOnOrAfter attribute specifies the time instant at which the validity interval  
510 has ended

511 The NotBefore and NotOnOrAfter attributes are optional. If the value is either  
512 omitted or equal to the start of the epoch it is unspecified. If the NotBefore attribute is  
513 unspecified the assertion is valid at any time before the time instant specified by the  
514 NotOnOrAfter attribute. If the NotOnOrAfter attribute is unspecified the assertion  
515 is valid from the time instant specified by the NotBefore attribute with no expiry. If  
516 neither attribute is specified the assertion is valid at any time.

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

519 Implementations MUST NOT generate time instances that specify leap seconds.

### 520 **1.7.2 Condition Type AudienceRestrictionConditionType**

- 521 • A <Condition> element of type  
522 AudienceRestrictionConditionType states that the assertion is  
523 addressed to one or more specific audience(s). Although a party that is outside the  
524 audience(s) specified is capable of drawing conclusions from an assertion, the  
525 issuer explicitly makes no representation as to accuracy or trustworthiness to such  
526 a party.

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

529 The following schema defines the AudienceRestrictionConditionType  
530 condition type:

```
531 <complexType name="AudienceRestrictionConditionType">  
532 <complexContent>  
533 <extension base="saml:AbstractConditionType">  
534 <sequence>  
535 <element name="Audience" type="xsd:uriReference"  
536 minOccurs="0" maxOccurs="unbounded"/>  
537 </sequence>  
538 </extension>  
539 </complexContent>  
540 </complexType>
```

## 541 **1.8 Advice**

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

544 1.8.1 Element <Advice>

545 The <Advice> element permits additional information to be included in an assertion  
546 that MAY be ignored by applications without affecting either the assertion semantics or  
547 validity. Advice elements MAY be specified in an extension schema. The advice element  
548 MAY be used to:

- 549 • Include evidence supporting the assertion claims to be cited, either directly  
550 (through incorporating the claims) or indirectly (by reference to the supporting  
551 assertions.
- 552 • State a proof of the assertion claims.
- 553 • Specify the timing and distribution points for updates to the assertion.

554 The following schema defines the <Advice> element:

```
555 <element name="Advice" type="saml:AdviceType"/>  
556 <complexType name="AdviceType">  
557 <sequence>  
558 <any namespace="##any" processContents="lax"  
559 minOccurs="0" maxOccurs="unbounded"/>  
560 </sequence>  
561 </complexType>  
562 </schema>
```

563 **1.9 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
AssertionAbstractType	Define new SAML Assertion
SubjectAssertionAbstractType	Define new SAML Assertion that takes a single SAML Subject as the subject.
AbstractConditionType	Define a new SAML Condition

568 In addition the <Advice> element permits arbitrary elements to be included without  
569 type restriction. An application is not required to understand or process any information  
570 contained within an <Advice> element however.[PHB7]



571 **2 SAML Protocol**

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

575 SAML aware clients may in addition use the request protocol defined by the  
 576 <Request> and <Response> elements described in this section. A <Request>  
 577 from the client is followed by a <Response> from the service Figure 1.



578

579 Figure 1: SAML Request/Response Protocol

580 **2.1 Namespaces**

581 The namespaces of the protocol schema are the same as those of the assertion schema  
 582 defined in section 1.1 with the addition of the namespace for the protocol schema itself.

583 `xmlns:samlp="http://www.oasis-`  
 584 `open.org/committees/security/docs/draft-sstc-schema-protocol-15.xsd"`

585 The following schema defines the XML namespaces for the assertion schema:

```

586 <?xml version="1.0" encoding="UTF-8"?>
587 <schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/protocol/"
588         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
589         xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
590         xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-
591 sstc-schema-assertion-15.xsd"
592         xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-
593 sstc-schema-protocol-15.xsd"
594         xmlns="http://www.w3.org/2000/10/XMLSchema"
595         elementFormDefault="unqualified">
596   <import namespace="http://www.oasis.org/tbs/1066-12-25/"
597           schemaLocation="draft-schema-assertion-10.xsd"/>
598   <import namespace="http://www.w3.org/2000/09/xmldsig#"
599           schemaLocation="xmldsig-core-schema.xsd"/>
600   <annotation>
601     <documentation>draft-schema-protocol-10.xsd</documentation>
602   </annotation>
  
```

603 **2.1.1 Basic Types**

604 The types defined in this section define XML types that are considered part of the schema  
 605 as a whole rather than a component of a particular element. This allows for greater  
 606 consistency and avoids the need for extension schemas to redefine the same types.

### 607 **2.1.1.1 Simple Type CompletenessSpecifierType**

608 The CompletenessSpecifierType type is used in a request to specify how a  
609 service should respond in cases where a client makes a request and the client is not  
610 authorized to receive part of the response. The CompletenessSpecifierType type  
611 defines two possible values "Any" and "All".

#### 612 **If Any is specified:**

613 The response contains the parts of the response that the client is authorized to  
614 receive.

#### 615 **If All is specified:**

616 The response is empty.

617 The following schema specifies the <CompletenessSpecifierType> type:

```
618 <simpleType name="CompletenessSpecifierType">  
619   <restriction base="string">  
620     <enumeration value="Any"/>  
621     <enumeration value="All"/>  
622   </restriction>  
623 </simpleType>
```

### 624 **2.1.1.2 Simple Type StatusCodeType**

625 The type StatusCodeType in a response specifies the status of the request. Four  
626 status values are defined:

#### 627 **Success**

628 The request succeeded.


#### 629 **Failure**

630 The request could not be performed by the service.

#### 631 **Error**

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

#### 633 **Unknown**

634  The request failed for unknown reasons[PHB8]

635 The following schema specifies the <StatusCodeType> type:

```
636 <simpleType name="StatusCodeType">  
637   <restriction base="string">  
638     <enumeration value="Success"/>  
639     <enumeration value="Failure"/>  
640     <enumeration value="Error"/>  
641     <enumeration value="Unknown"/>  
642   </restriction>  
643 </simpleType>
```

## 644 **2.2 Request**

### 645 **2.2.1 Abstract Type RequestAbstractType**

646 All SAML requests are extensions of the RequestAbstractType abstract type. The  
647 RequestAbstractType requires that all SAML requests specify the version number  
648 of the SAML protocol and a request identifier.

649 The following schema defines the RequestAbstractType abstract type:

```
650 <complexType name="RequestAbstractType" abstract="true">  
651   <attribute name="RequestID" type="saml:IDType" use="required"/>  
652   <attribute name="Version" type="string" use="required"/>  
653 </complexType>
```

#### 654 **2.2.1.1 Attribute RequestID**

655 The RequestID attribute defines a unique identifier for the assertion request. The  
656 RequestID element in a request MUST match the InResponseTo element in the  
657 corresponding response.

#### 658 **2.2.1.2 Attribute Version**

659 Each request MUST specify the SAML protocol version identifier. The identifier for this  
660 version of SAML is the string "1.0".

### 661 **2.2.2 Element <Request>**

662 The <Request> element specifies a SAML request. This may contain either a query or  
663 a request for a specific assertion identified by AssertionID.

664 The following schema defines the <Request> element:

```
665 <element name="Request" type="samlp:RequestType"/>  
666 <complexType name="RequestType">  
667   <complexContent>  
668     <extension base="samlp:RequestAbstractType">  
669       <choice>  
670         <element name="Query" type="samlp:QueryType"/>  
671         <element ref="saml:AssertionID" maxOccurs="unbounded"/>  
672       </choice>  
673     </extension>  
674   </complexContent>  
675 </complexType>
```

### 676 **2.2.3 Abstract Type QueryAbstractType**

677 The QueryAbstractType abstract type is the base type from which all SAML  
678 request query elements are derived. The abstract type contains no elements or attributes.

679 The following schema defines the QueryAbstractType abstract type:

```
680 <complexType name="QueryAbstractType" abstract="true"/>
```

## 681 2.2.4 Abstract Type SubjectQueryAbstractType

682 The SubjectQueryAbstractType type extends the Query type to specify a query  
683 with a specific subject as its principal.

684 The following schema defines the SubjectQueryAbstractType abstract type:

```
685 <complexType name="SubjectQueryAbstractType" abstract="true">  
686 <complexContent>  
687 <extension base="samlp:QueryAbstractType">  
688 <sequence>  
689 <element ref="saml:Subject"/>  
690 </sequence>  
691 </extension>  
692 </complexContent>  
693 </complexType>
```

## 694 2.3 Authentication Query

695 The AuthenticationQueryType makes the query “What authentication assertions are  
696 available for this Subject?”

697 The response will be in the form of an Authentication assertion.

### 698 2.3.1 Subject Query Type AuthenticationQueryType

699 An AuthenticationQuery contains all the elements and attributes of a  
700 SubjectQuery and extends them as follows:

701 **<AuthenticationMethod>** [Optional]  
702 The <AuthenticationMethod> element if present can be used to as a filter  
703 for possible responses. This supports the query “What authentication assertions do  
704 you have for this Subject with the following AuthenticationMethod?”

705 A SAML processor will return a certain number of Authentication Assertions in response  
706 to a Query of type AuthenticationQueryType. The <Subject> of the returned  
707 assertions MUST be identical to the <Subject> element of the Query. If the  
708 <AuthenticationMethod> field is present in the Query at least one  
709 <AuthenticationMethod> field in the response MUST match. There is no  
710 implication that all such assertions must be returned.

711 The following schema defines the AuthenticationQueryType type:

```
712 <complexType name="AuthenticationQueryType">  
713 <complexContent>  
714 <extension base="samlp:SubjectQueryAbstractType">  
715 <sequence>  
716 <element ref="saml:AuthenticationMethod" minOccurs="0"/>  
717 <!--do we want more than one of these?-->  
718 </sequence>  
719 </extension>  
720 </complexContent>  
721 </complexType>
```

## 722 **2.4 Attribute Query**

723 An Attribute Query makes the query “Return the requested attributes for this Subject”

724 The response will be in the form of an Attribute Assertion.

### 725 **2.4.1 Subject Query Type AttributeQueryType**

726 An AttributeQueryType contains all the elements and attributes of a  
727 SubjectQueryAbstractType and extends them as follows:

728 **<Attribute>** [Any number]

729 Each <Attribute> element specifies an attribute that is to be returned. If no  
730 attributes are specified the scope of the query is implicit.

731 **<CompletenessSpecifier>** [Required]

732 The <CompletenessSpecifier> element specifies the desired behavior in  
733 the case that access to some of the requested attributes is not authorized using the  
734 CompletenessSpecifier.

735 The following schema defines the <AttributeQueryType> type:

```
736 <complexType name="AttributeQueryType">  
737   <complexContent>  
738     <extension base="samlp:SubjectQueryAbstractType">  
739       <sequence>  
740         <element ref="saml:AttributeNamespaceName"  
741           minOccurs="0" maxOccurs="unbounded"/>  
742         <element name="CompletenessSpecifier"  
743           type="samlp:CompletenessSpecifierType" default="All"/>  
744       </sequence>  
745     </extension>  
746   </complexContent>  
747 </complexType>
```

## 748 **2.5 Authorization Query**

749 An Authorization Query makes the query : “Should action(s) Y on resource Z be allowed  
750 for subject S given evidence E?”

751 The answer comes in the form of an Authorization Decision assertion. The action(s) and  
752 resource are optionally namespace-scoped..

### 753 **2.5.1 Subject Query Type AuthorizationQueryType**

754 An AuthorizationQuery contains all the elements and attributes of a  
755 SubjectQueryAbstractType and extends them as follows:

756 **<Object>** [Required]

757 The <Object> element specifies the resource and action(s) for which  
758 authorization is requested.

759        **<Evidence>** [Any number]  
760            Each **<Evidence>** element specifies an assertion that the service may rely upon  
761            in making its response.

762    The following schema defines the `AuthorizationQueryType` type:

```
763        <element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>  
764        <complexType name="AuthorizationQueryType">  
765            <complexContent>  
766                <extension base="samlp:SubjectQueryAbstractType">  
767                    <sequence>  
768                        <element ref="saml:Evidence"  
769                            minOccurs="0" maxOccurs="unbounded"/>  
770                        <element ref="saml:Object"/>  
771                    </sequence>  
772                </extension>  
773            </complexContent>  
774        </complexType>
```

## 775    **2.6    Response**

### 776    **2.6.1    Abstract Type ResponseAbstractType**

777    The response to a Request is of a type that extends the `ResponseAbstractType`  
778    abstract type. The `ResponseAbstractType` specifies information that is common to  
779    all SAML responses, the SAML protocol version, the identifier of the response and the  
780    identifier of the request that is responded to.

781    The following schema defines the `ResponseAbstractType` abstract type:

```
782        <complexType name="ResponseAbstractType" abstract="true">  
783            <attribute name="ResponseID" type="saml:IDType" use="required"/>  
784            <attribute name="InResponseTo" type="saml:IDType" use="required"/>  
785            <attribute name="Version" type="string" use="required"/>  
786        </complexType>
```

#### 787    **2.6.1.1    Attribute ResponseID**

788    The `ResponseID` attribute specifies an identifier of type `IDType` for the response.

#### 789    **2.6.1.2    Attribute InResponseTo**

790    The `ResponseID` attribute specifies the identifier of type `IDType` specified in the  
791    `RequestID` attribute in the SAML Request to which it is a response.

#### 792    **2.6.1.3    Attribute Version**

793    Each response **MUST** specify the SAML version identifier. The identifier for this version  
794    of SAML is the string "1.0".[PHB9]



## 795 2.6.2 Element <Response>

796 The <Response> element extends the ResponseAbstractType and specifies  
797 the status of the corresponding SAML Request and a list of zero or more assertions that  
798 answer the request.

799 The following schema defines the <Response> element:


```
800 <element name="Response" type="samlp:ResponseType"/>  
801 <complexType name="ResponseType">  
802 <complexContent>  
803 <extension base="samlp:ResponseAbstractType">  
804 <sequence>  
805 <element ref="saml:Assertion"  
806 minOccurs="0" maxOccurs="unbounded"/>  
807 </sequence>  
808 <attribute name="StatusCode" type="samlp:StatusCodeType"  
809 use="required"/>  
810 </extension>  
811 </complexContent>  
812 </complexType>  
813 </schema>
```

## 814 2.7 Schema Extension

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

818 The following abstract types are defined in the schema:[PHB10]

---

Abstract Type	Purpose
RequestAbstractType	Specify a new SAML request other than a query.
QueryAbstractType	Specify a new SAML request that is a query.
 SubjectQueryAbstractType	Specify a new SAML request that is a query concerning a single subject.
ResponseAbstractType	Specify a new SAML response.

---

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

### 821 3 References

- 822 [Kerberos] TBS
- 823 [SAML-USE] TBS
- 824 [PKCS1] Kaliski, B., *PKCS #1: RSA Encryption Version 2.0*, RSA  
825 Laboratories, also IETF RFC 2437, October 1998.
- 826 [RFC-2104] Krawczyk, H., Bellare, M. and R. Canetti, *HMAC: Keyed Hashing*  
827 *for Message Authentication*, IETF RFC 2104, February 1997.
- 828 [SOAP] D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn,  
829 H. Frystyk Nielsen, S Thatte, D. Winer. *Simple Object Access*  
830 *Protocol (SOAP) 1.1*, W3C Note 08 May 2000,  
831 <http://www.w3.org/TR/SOAP>
- 832 [WSSL] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web*  
833 *Services Description Language (WSDL) 1.0* September 25, 2000,  
834 <http://msdn.microsoft.com/xml/general/wSDL.asp>
- 835 [XACML] TBS
- 836 [XTASS] P. Hallam-Baker, *XML Trust Axiom Service Specification 1.0*,  
837 VeriSign Inc. January 2001. <http://www.xmltrustcenter.org/>
- 838 [XML-SIG] D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon.  
839 *XML-Signature Syntax and Processing*, World Wide Web  
840 Consortium. <http://www.w3.org/TR/xmlsig-core/>
- 841 [XML-SIG-XSD] XML Signature Schema available from  
842 [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)  
843 [core-schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).
- 844 [XML-Enc] *XML Encryption Specification*, In development.
- 845 [XML-Schema1] H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn. *XML*  
846 *Schema Part 1: Structures*, W3C Working Draft 22 September  
847 2000, <http://www.w3.org/TR/2000/WD-xmlschema-1-20000922/>,  
848 latest draft at <http://www.w3.org/TR/xmlschema-1/>
- 849 [XML-Schema2] P. V. Biron, A. Malhotra, *XML Schema Part 2: Datatypes*; W3C  
850 Working Draft 22 September 2000,  
851 <http://www.w3.org/TR/2000/WD-xmlschema-2-20000922/>, latest  
852 draft at <http://www.w3.org/TR/xmlschema-2/>



853 **4 Identifiers**

854 **4.1 Authentication Protocol Identifiers**

855 4.1.1 SAML Artifact

856 4.1.2 Assertion Bearer

857 4.1.3 User Name and Password (Pass-through)

858 4.1.4 User Name and Password (One-Way-Function SHA-1)

859 4.1.5 Kerberos

860 4.1.6 SSL/TLS Certificate Based Client Authentication

861 4.1.7 Object Authenticator (SHA-1)

862 **4.2 Action Identifiers**

863 4.2.1 Read/Write/Execute/Delete/Control

864 4.2.2 Read/Write/Execute/Delete/Control with Negation

865 4.2.3 Get/Head/Put/Post

866 4.2.4 UNIX file Permissions

867 **5 Appendix**

868 **5.1 Assertion Schema**

869 [See separate file]

870 **5.2 Protocol Schema**

871 [See separate file]

Page: 5

[PHB1] Insert the class diagram here

Page: 5

[PHB2]This schema is actually to the previous version since that is what Spy 3.5 accepts.

Page: 7

[PHB3]Here we need to redo the diagram to express the inheritance mechanism we choose in the end.

Page: 8

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

Page: 10

[PHB5]Yikes! maybe we should put an IDType in here so that the security domain is at least unique!

Page: 14

[PHB6]But the current definition does not give a ground case, how about a string?

Page: 16

[PHB7]Add in here the substitution group issue.

Page: 18

[PHB8]Need to have text for these, how exactly does failure differ from error?

Page: 22

[PHB9] Other options here include specifying the highest version number that the server can supply. I suspect however that in the web services context that is unnecessary since it will be taken care of by WSDL.

Page: 23

[PHB10]If we use substitution groups add in the text to specify the requirement