

1

•
•
•
•
•
•
•

2 **Security Assertions Markup Language**_[pm1]

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 0.12: July 27th 2001*

18

19

19

20 Security Assertions Markup Language

21 Version 0.12

22 Table Of Contents

23	Table Of Contents	2
24	1 XML Assertion Syntax	4
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 SubjectAssertionType	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	13
38	1.6.1 Assertion Type AttributeAssertionType	13
39	1.7 Conditions	14
40	1.7.1 Element <Conditions>	14
41	1.7.2 Condition Type AudienceRestrictionConditionType	15
42	1.8 Advice	16
43	1.8.1 Element <Advice>	16
44	1.9 Schema Extension	16
45	2 SAML Protocol	17
46	2.1 Namespaces	17
47	2.1.1 Basic Types	17
48	2.2 Request	18
49	2.2.1 Abstract Type SAMLAbstractRequestType	18
50	2.2.2 Element <SAMLRequest>	19
51	2.2.3 Abstract Type SAMLQueryType	19
52	2.2.4 Abstract Type SubjectQueryType	19
53	2.3 Authentication Query	20
54	2.3.1 Subject Query Type AuthenticationQueryType	20
55	2.4 Attribute Query	20
56	2.4.1 Subject Query Type SAMLAttributeQueryType	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 <code>SAMLAbstractResponseType</code>	22
61	2.6.2	Element <code><SAMLResponse></code>	22
62	2.7	Schema Extension	23
63	3	References	24
64	4	Identifiers	25
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	4.2	Action Identifiers	25
73	4.2.1	Read/Write/Execute/Delete/Control	25
74	4.2.2	Read/Write/Execute/Delete/Control with Negation	25
75	4.2.3	Get/Head/Put/Post	25
76	4.2.4	UNIX file Permissions	25
77	5	Appendix	26
78	5.1	Assertion Schema	26
79	5.2	Protocol Schema	29
80			

81 **1 XML Assertion Syntax**

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

83 **Authentication Assertion**

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

86 **Authorization Decision Assertion**

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

90 **Attribute Assertion**

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

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

95 **Basic Information.**

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

99 **The Asserted Statement**

100 **The statement that is asserted by the issuer of the assertion.**

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

105 **Conditions.**

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

110 **Advice.**

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

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


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

118  [Class diagram to be inserted][PHB2]


119 1.1 Namespaces

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

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

124  `xmlns="http://www.w3.org/2001/XMLSchema"`
125 `xmlns:xsd="http://www.w3.org/2001/XMLSchema"` [PHB3]

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

128  `xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"` [PHB4]

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

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

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

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

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

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

149 1.1.1 Basic Types

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

153 **1.1.1.1 Simple Type IDType**

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

157 IDType identifiers MUST satisfy the following properties:

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

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

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

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

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

```
173 <element name="AssertionID" type="saml:IDType"/>  
174 <simpleType name="IDType">  
175 <restriction base="string"/>  
176 </simpleType>
```

177 **1.1.1.2 Simple Type DecisionType**

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

180 **Permit**

181 The specified action is permitted.

182 **Deny**

183 The specified action is denied.

184 **Indeterminate**


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

186 The following schema specifies the DecisionType type:

```
187 <simpleType name="DecisionType">  
188 <restriction base="string">
```

```
189     <enumeration value="Permit"/>
190     <enumeration value="Deny"/>
191     <enumeration value="Indeterminate"/>
192   </restriction>
193 </simpleType>
```

194 1.2 SAML Assertion

195 A SAML Assertion is specified by a single XML element whose type is derived from the
196 abstract XML type `AssertionType`.^[PHB5] The abstract `AssertionType` specifies
197 the basic information that is common to all SAML assertions. Instances of SAML
198 assertions have concrete types that are extensions of the base `AssertionType`. 

199 1.2.1 Abstract Element `<Assertion>`

200 The element `<Assertion>` of abstract type `AssertionType` is used to specify a
201 SAML assertion.

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

```
204   <Assertion xsiType=AttributeAssertionType>
```

205 The following schema specifies the `<Assertion>` element and `AssertionType`
206 abstract type:

```
207   <element name="Assertion" type="saml:AssertionType"/>
208   <complexType name="AssertionType" abstract="true">
209     <sequence>
210       <element name="Conditions" type="saml:ConditionsType" minOccurs="0"/>
211       <element name="Advice" type="saml:AdviceType" minOccurs="0"/>
212     </sequence>
213     <attribute name="Version" type="string" use="required"/>
214     <attribute name="AssertionID" type="saml:IDType" use="required"/>
215     <attribute name="Issuer" type="string" use="required"/>
216     <attribute name="IssueInstant" type="timeInstant" use="required"/>
217   </complexType>
```

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


221 1.2.1.1 Attribute Version

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

224 1.2.1.2 Attribute AssertionID

225 The `AssertionID` attribute specifies the assertion identifier.

226 1.2.1.3 Attribute Issuer

227 The `Issuer` attribute specifies the issuer of the assertion by means of a string^[PHB6]. 

228 **1.2.1.4 Attribute IssueInstant**

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

231 **1.2.2 Element <AssertionSpecifier>**

232 The <AssertionSpecifier> element specifies an assertion either by reference or
233 by value. An assertion is specified by reference to the value of its AssertionID attribute.
234 An assertion is specified by value by including the entire assertion.

235 The following schema specifies the <AssertionSpecifier> element:

```
236 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>  
237 <complexType name="AssertionSpecifierType">  
238   <choice>  
239     <element ref="saml:AssertionID" />  
240     <element ref="saml:Assertion"/>  
241   </choice>  
242 </complexType>
```

243 **1.3 Subject Assertion**

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

246 **1.3.1 Abstract Type SubjectAssertionType**

247 The SubjectAssertionType type is an abstract type that extends the
248 AssertionType to include a <Subject> element.

249 The following schema defines the SubjectAssertionType abstract type:

```
250 <complexType name="SubjectAssertionType" abstract="true">  
251   <complexContent>  
252     <extension base="saml:AssertionType">  
253       <sequence>  
254         <element ref="saml:Subject"/>  
255       </sequence>  
256     </extension>  
257   </complexContent>  
258 </complexType>
```

259 **1.3.2 Element <Subject>**

260 The <Subject> element specifies a party by one of the following means:

- 261 • A name.
- 262 • By information that allows the party to be authenticated.
- 263 • By reference to another assertion or by containment of another assertion.

264 If a <Subject> element contains more than one subject specification the issuer is
265 asserting that all the subject specifications present specify the same subject. For example
266 if both a <NameIdentifier> and a <Authenticator> element are present the
267 issuer is asserting that the authentication data authenticates the party with the specified
268 name.

269 The following schema defines the <Subject> element:

```
270 <element name="Subject" type="saml:SubjectType"/>
271 <complexType name="SubjectType">
272   <choice maxOccurs="unbounded">
273     <element ref="saml:NameIdentifier"
274       minOccurs="0" maxOccurs="unbounded"/>
275     <element ref="saml:Authenticator"
276       minOccurs="0" maxOccurs="unbounded"/>
277     <element ref="saml:AssertionSpecifier"
278       minOccurs="0" maxOccurs="unbounded"/>
279   </choice>
280 </complexType>
```

281 1.3.2.1 Element <Authenticator>

282 The <Authenticator> element specifies a subject by specifying data that authenticates the
283 subject.

284 <Protocol> [Required]

285 Each <Protocol> element specifies a URI that identify a protocol that may be
286 used to authenticate the subject.

287 <AuthData> [Optional]

288 Each <AuthData> element specifies additional authentication information used
289 by a specific authentication protocol.

290 <ds:KeyInfo> [Optional]

291 An XML Signature <ds:KeyInfo> element that specifies a cryptographic key
292 held by the subject.


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

294 The following schema defines the <Authenticator> element:

```
295 <element name="Authenticator" type="saml:AuthenticatorType"/>
296 <complexType name="AuthenticatorType">
297   <sequence>
298     <element name="Protocol" type="uriReference"
299       maxOccurs="unbounded"/>
300     <element name="Authdata" type="string" minOccurs="0"/>
301     <element ref="ds:KeyInfo" minOccurs="0"/>
302   </sequence>
303 </complexType>
```

304 1.3.2.2 Element <NameIdentifier>

305 The NameIdentifier type specifies a subject by a combination of a name and a security
306 domain.

307 The interpretation of the security domain and the name are left to individual
308 implementations.[PHB7] 

309 The following schema defines the <NameIdentifier> element:

```
310 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
311 <complexType name="NameIdentifierType">
312   <sequence>
313     <element name="SecurityDomain" type="string"/>
314     <element name="Name" type="string"/>
315   </sequence>
316 </complexType>
```

317 1.4 Authentication Assertion

318 An authentication assertion is an assertion by the issuer that the subject was authenticated
319 by a particular means at a particular time.[PHB8] 

320 1.4.1 Assertion Type AuthenticationAssertionType

321 The AuthenticationAssertionType extends the SubjectAssertionType
322 with the addition of the following elements:

323 <AuthenticationCode> [Required]

324 The <AuthenticationCode> element specifies the type of Authentication
325 that took place.

326 <AuthenticationInstant> [Required]

327 The <AuthenticationInstant> element specifies the time at which the
328 authentication took place.


329 <AuthenticationLocale> [Optional]

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

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

```
333 <complexType name="AuthenticationAssertionType">
334   <complexContent>
335     <extension base="saml:SubjectAssertionType">
336       <sequence>
337         <element ref="saml:AuthenticationCode"/>
338         <element name="AuthenticationInstant" type="timeInstant"/>
339         <element name="AuthLocale"
340           type="saml:AuthLocaleType" minOccurs="0"/>
341       </sequence>
342     </extension>
343   </complexContent>
344 </complexType>
```

345 1.4.1.1 Element <AuthenticationCode>

346 The <AuthenticationCode> element specifies the type of Authentication that took
347 place.[PHB9] 

348 The following schema defines the <AuthenticationCode> element:

```
349 <element name="AuthenticationCode" type="saml:AuthenticationCodeType"/>
350 <simpleType name="AuthenticationCodeType">
351   <restriction base="string"/>
352 </simpleType>
```

353 1.4.1.2 Type AuthLocale

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

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

358 The following schema defines the <AuthLocale> type:

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

365 1.5 Authorization Decision Assertion

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

369 1.5.1 Assertion Type AuthorizationDecisionAssertionType

370 The AuthorizationDecisionAssertionType extends the
371 SubjectAssertionType with the addition of the following elements:

372 <Object> [Required]

373 The <Object> element specifies a set of actions on a specified resource

374 <Answer> [Required]

375 The <Answer> element specifies the decision with respect to the specified
376 object.

377 <Evidence> [Optional]

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

380 The following schema defines the <AuthorizationDecisionAssertionType >
381 type:

```
382 <complexType name="AuthorizationDecisionAssertionType">
383   <complexContent>
384     <extension base="saml:SubjectAssertionType">
385       <sequence>
386         <element ref="saml:Object"/>
```

```
387         <element name="Answer" type="saml:DecisionType"/>
388         <element name="saml:Evidence"
389             minOccurs="0" maxOccurs="unbounded"/>
390     </sequence>
391 </extension>
392 </complexContent>
393 </complexType>
```

394 1.5.1.1 Element <Object>

395 The <Object> element specifies an authorization object that consists of a set of actions
396 on a specified resource. The <Object> element contains the following elements:

397 <Resource> [Required]

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

399 <Namespace> [Optional]

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

402 <Action> [One or more]

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

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

406 The following schema defines the <Object> element:

```
407 <element name="Object" type="saml:ObjectType"/>
408 <complexType name="ObjectType">
409     <sequence>
410         <element name="Resource" type="xsd:uriReference"/>
411         <element name="Namespace" type="uriReference" minOccurs="0"/>
412         <element name="Action" type="string" maxOccurs="unbounded"/>
413     </sequence>
414 </complexType>
```

415 1.5.1.2 Element <Evidence>

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

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

422 The following schema defines the <Evidence> element:

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

424 **1.6 Attribute Assertion**

425 An attribute assertion asserts that the specified subject is associated with the specified
426 attribute(s)

427 **1.6.1 Assertion Type AttributeAssertionType**

428 The AttributeAssertionType extends the SubjectAssertionType with the
429 addition of the following element:

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

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

432 The following schema defines the AttributeAssertionType assertion type:

```
433 <complexType name="AttributeAssertionType">  
434   <complexContent>  
435     <extension base="saml:SubjectAssertionType">  
436       <sequence>  
437         <element ref="saml:Attribute" maxOccurs="unbounded"/>  
438       </sequence>  
439     </extension>  
440   </complexContent>  
441 </complexType>
```

442 **1.6.1.1 Element <Attribute>**

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

446 **<AttributeNameSpace>** [Optional]

447 The <AttributeNameSpace> element specifies the namespace in which the
448 <AttributeName> elements are interpreted.

449 **<AttributeName>** [Required]

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

451 **<AttributeValue>** [Any number]

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

453 If no <AttributeNameSpace> element is specified the interpretation of
454 <AttributeName> elements is left to the implementation.

455 The following schema defines the <Attribute> element:

```
456 <element name="Attribute" type="saml:AttributeType"/>  
457 <complexType name="AttributeType">  
458   <sequence>  
459     <element name="AttributeName" type="string"/>  
460     <element name="AttributeNameSpace"  
461       type="uriReference" minOccurs="0"/>  
462     <element name="AttributeValue" type="saml:AttributeValueType"  
463       minOccurs="0" maxOccurs="unbounded"/>
```

```
464     </sequence>
465   </complexType>
```

466 1.6.1.2 Type AttributeValueType

467 An <AttributeValue> element is of type AttributeValueType. The
468 AttributeValueType allows the inclusion of any element in any namespace.[PHB10]



469 The following schema defines the AttributeValueType type:

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

476 1.7 Conditions

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

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

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

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

488 1.7.1 Element <Conditions>

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

493 The following schema defines the <Conditions> element:

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

504 **1.7.1.1 Attributes NotBefore and NotOnOrAfter**

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

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

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

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

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

519 **1.7.2 Condition Type AudienceRestrictionConditionType**

- 520
- 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
a party.

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

528

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
AssertionType	Define new SAML Assertion
SubjectAssertionType	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.^[PHB11]



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 <SAMLRequest> and <SAMLResponse> elements described in this section. A
 577 <SAMLRequest> from the client is followed by a <SAMLResponse> from the service
 578 Figure 1.



579

580 Figure 1: SAML Request/Response Protocol

581 **2.1 Namespaces**

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

584 `xmlns:samlp=http://www.oasis.org/tbs/1066-12-25/protocol/`

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.org/tbs/1066-12-25/"
591         xmlns:samlp="http://www.oasis.org/tbs/1066-12-25/protocol/"
592         xmlns="http://www.w3.org/2000/10/XMLSchema"
593         elementFormDefault="unqualified">
594   <import namespace="http://www.oasis.org/tbs/1066-12-25/"
595           schemaLocation="draft-schema-assertion-10.xsd"/>
596   <import namespace="http://www.w3.org/2000/09/xmldsig#"
597           schemaLocation="xmldsig-core-schema.xsd"/>
598   <annotation>
599     <documentation>draft-schema-protocol-10.xsd</documentation>
600   </annotation>
  
```

601 **2.1.1 Basic Types**

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

605 **2.1.1.1 Simple Type CompletenessSpecifierType**

606 The CompletenessSpecifierType type is used in a request to specify how a
 607 service should respond in cases where a client makes a request and the client is not

608 authorized to receive part of the response. The `CompletenessSpecifierType` type
609 defines two possible values "Any" and "All".

610 **If Any is specified:**

611 The response contains the parts of the response that the client is authorized to
612 receive.

613 **If All is specified:**

614 The response is empty.


615 The following schema specifies the `<CompletenessSpecifierType>` type:

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

622 **2.1.1.2 Simple Type `StatusCodeType`**

623 The type `StatusCodeType` in a response specifies the status of the request. Four
624 status values are defined:

625 **Success**

626 The request succeeded. 

627 **Failure**

628 The request could not be performed by the service.

629 **Error**

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

631 **Unknown**


632 The request failed for unknown reasons^[PHB12]

633 The following schema specifies the `<StatusCodeType>` type:

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

642 **2.2 Request**

643 **2.2.1 Abstract Type `SAMLAbstractRequestType`**

644 ^[PHB13]All SAML requests are extensions of the `SAMLAbstractRequestType`
645 abstract type. The `SAMLAbstractRequestType` requires that all SAML requests
646 specify the version number of the SAML protocol and a request identifier. 

647 The following schema defines the `SAMLAbstractRequestType` abstract type:

```
648 <complexType name="SAMLAbstractRequestType" abstract="true">
649   <attribute name="RequestID" type="saml:IDType" use="required"/>
650   <attribute name="Version" type="string" use="required"/>
651 </complexType>
```

652 2.2.1.1 Attribute RequestID

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

656 2.2.1.2 Attribute Version

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


659 2.2.2 Element <SAMLRequest>

660 The <SAMLRequest> element specifies a SAML request. This may contain either a
661 query or a request for a specific assertion identified by AssertionID.

662 The following schema defines the <SAMLRequest> element:

```
663 <element name="SAMLRequest" type="samlp:SAMLRequestType"/>
664 <complexType name="SAMLRequestType">
665   <complexContent>
666     <extension base="samlp:SAMLAbstractRequestType">
667       <choice>
668         <element name="Query" type="samlp:SAMLQueryType"/>
669         <element ref="saml:AssertionID" maxOccurs="unbounded"/>
670       </choice>
671     </extension>
672   </complexContent>
673 </complexType>
```


674 2.2.3 Abstract Type SAMLQueryType

675  [PHB14] The SAMLQueryType abstract type is the base type from which all SAML
676 request query elements are derived. The abstract type contains no elements or attributes.

677 The following schema defines the SAMLQueryType abstract type:

```
678 <complexType name="SAMLQueryType" abstract="true"/>
```

679 2.2.4 Abstract Type SubjectQueryType

680  [PHB15] The SubjectQueryType type extends the SAMLQuery type to specify a query
681 with a specific subject as its principal.

682 The following schema defines the SubjectQueryType abstract type:

```
683 <complexType name="SubjectQueryType" abstract="true">
684   <complexContent>
685     <extension base="samlp:SAMLQueryType">
```

```
686         <sequence>
687             <element ref="saml:Subject"/>
688         </sequence>
689     </extension>
690 </complexContent>
691 </complexType>
```

692 2.3 Authentication Query

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

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

696 2.3.1 Subject Query Type AuthenticationQueryType

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

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

703 A SAML processor will return a certain number of Authentication Assertions in response
704 to a SAMLQuery of type AuthenticationQueryType. The <Subject> and
705 <AuthenticationCode> of the returned assertions MUST be identical to the subject
706 (and optional <AuthenticationCode>) fields of the SAMLQuery. There is no
707 implication that all such assertions must be returned.

708 The following schema defines the AuthenticationQueryType type:

```
709 <complexType name="AuthenticationQueryType">
710     <complexContent>
711         <extension base="samlp:SubjectQueryType">
712             <sequence>
713                 <element ref="saml:AuthenticationCode" minOccurs="0"/>
714                 <!--do we want more than one of these?-->
715             </sequence>
716         </extension>
717     </complexContent>
718 </complexType>
```

719 2.4 Attribute Query

720 An Attribute Query makes the query “Return all of the attributes for this Subject (that I
721 am allowed to see)?”

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

723 2.4.1 Subject Query Type SAMLAttributeQueryType

724 An SAMLAttributeQueryType contains all the elements and attributes of a
725 SubjectQueryType and extends them as follows:

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

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

729 **<CompletenessSpecifier>** [Required]

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

733

734 The following schema defines the <SAMLAttributeQueryType> type:

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

747 2.5 Authorization Query

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

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

752 2.5.1 Subject Query Type AuthorizationQueryType

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

755 **<Object>** [Required]

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

758 **<Evidence>** [Any number]

759 Each <Evidence> element specifies an assertion that the service may rely upon
760 in making its response.

761

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:SubjectQueryType">
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 SAMLAbstractResponseType

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

782 The following schema defines the SAMLAbstractResponseType abstract type:

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


788 2.6.1.1 Attribute ResponseID

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

790 2.6.1.2 Attribute InResponseTo

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

793 2.6.1.3 Attribute Version

794 Each response MUST specify the SAML version identifier. The identifier for this version
795 of SAML is the string "1.0".[PHB16] 

796 2.6.2 Element <SAMLResponse>


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

800 The following schema defines the <SAMLResponse> element:

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

815 2.7 Schema Extension

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

819 The following abstract types are defined in the schema:[PHB17] 

Abstract Type	Purpose
SAMLAbstractRequestType	Specify a new SAML request other than a query.
SAMLQueryType	Specify a new SAML request that is a query.
SubjectQueryType	Specify a new SAML request that is a query concerning a single subject.
SAMLAbstractResponseType	Specify a new SAML response.

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

822 3 References

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

854 **4 Identifiers**

855 **4.1 Authentication Protocol Identifiers**

856 4.1.1 SAML Artifact

857 4.1.2 Assertion Bearer

858 4.1.3 User Name and Password (Pass-through)

859 4.1.4 User Name and Password (One-Way-Function SHA-1)

860 4.1.5 Kerberos

861 4.1.6 SSL/TLS Certificate Based Client Authentication

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 <?xml version="1.0" encoding="UTF-8"?>
870 <schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/"
871       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
872       xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
873       xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"
874       xmlns="http://www.w3.org/2000/10/XMLSchema"
875       elementFormDefault="unqualified">
876   <import namespace=" http://www.w3.org/2000/09/xmldsig#"
877         schemaLocation="xmldsig-core-schema.xsd"/>
878   <annotation>
879     <documentation>draft-schema-consensus-10.xsd</documentation>
880   </annotation>
881
882   <element name="AssertionID" type="saml:IDType"/>
883   <simpleType name="IDType">
884     <restriction base="string"/>
885   </simpleType>
886   <simpleType name="DecisionType">
887     <restriction base="string">
888       <enumeration value="Permit"/>
889       <enumeration value="Deny"/>
890       <enumeration value="Indeterminate"/>
891     </restriction>
892   </simpleType>
893
894   <element name="Assertion" type="saml:AssertionType" />
895   <complexType name="AssertionType" abstract="true">
896     <sequence>
897       <element name="Conditions" type="saml:ConditionsType" minOccurs="0"/>
898       <element name="Advice" type="saml:AdviceType" minOccurs="0"/>
899     </sequence>
900     <attribute name="Version" type="string" use="required"/>
901     <attribute name="AssertionID" type="saml:IDType" use="required"/>
902     <attribute name="Issuer" type="string" use="required"/>
903     <attribute name="IssueInstant" type="timeInstant" use="required"/>
904   </complexType>
905
906   <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
907   <complexType name="AssertionSpecifierType">
908     <choice>
909       <element ref="saml:AssertionID" />
910       <element ref="saml:Assertion"/>
911     </choice>
912   </complexType>
913
914   <complexType name="SubjectAssertionType" abstract="true">
915     <complexContent>
916       <extension base="saml:AssertionType">
917         <sequence>
918           <element ref="saml:Subject"/>
919         </sequence>
920       </extension>
921     </complexContent>
922   </complexType>
923
924   <element name="Subject" type="saml:SubjectType"/>
925   <complexType name="SubjectType">

```

```

926     <choice maxOccurs="unbounded">
927         <element ref="saml:NameIdentifier"
928             minOccurs="0" maxOccurs="unbounded"/>
929         <element ref="saml:Authenticator"
930             minOccurs="0" maxOccurs="unbounded"/>
931         <element ref="saml:AssertionSpecifier"
932             minOccurs="0" maxOccurs="unbounded"/>
933     </choice>
934 </complexType>
935
936 <element name="Authenticator" type="saml:AuthenticatorType"/>
937 <complexType name="AuthenticatorType">
938     <sequence>
939         <element name="Protocol" type="uriReference"
940             maxOccurs="unbounded"/>
941         <element name="Authdata" type="string" minOccurs="0"/>
942         <element ref="ds:KeyInfo" minOccurs="0"/>
943     </sequence>
944 </complexType>
945
946 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
947 <complexType name="NameIdentifierType">
948     <sequence>
949         <element name="SecurityDomain" type="string"/>
950         <element name="Name" type="string"/>
951     </sequence>
952 </complexType>
953
954 <complexType name="AuthenticationAssertionType">
955     <complexContent>
956         <extension base="saml:SubjectAssertionType">
957             <sequence>
958                 <element ref="saml:AuthenticationCode"/>
959                 <element name="AuthenticationInstant" type="timeInstant"/>
960                 <element name="AuthLocale"
961                     type="saml:AuthLocaleType" minOccurs="0"/>
962             </sequence>
963         </extension>
964     </complexContent>
965 </complexType>
966
967 <element name="AuthenticationCode" type="saml:AuthenticationCodeType"/>
968 <simpleType name="AuthenticationCodeType">
969     <restriction base="string"/>
970 </simpleType>
971
972 <complexType name="AuthLocaleType">
973     <sequence>
974         <element name="IP" type="string" minOccurs="0"/>
975         <element name="DNS_Domain" type="string" minOccurs="0"/>
976     </sequence>
977 </complexType>
978
979 <complexType name="AuthorizationDecisionAssertionType">
980     <complexContent>
981         <extension base="saml:SubjectAssertionType">
982             <sequence>
983                 <element ref="saml:Object"/>
984                 <element name="Answer" type="saml:DecisionType"/>
985                 <element name="saml:Evidence"
986                     minOccurs="0" maxOccurs="unbounded"/>
987             </sequence>
988         </extension>

```

```

989     </complexContent>
990 </complexType>
991
992 <element name="Object" type="saml:ObjectType"/>
993 <complexType name="ObjectType">
994   <sequence>
995     <element name="Resource" type="xsd:uriReference"/>
996     <element name="Namespace" type="uriReference" minOccurs="0"/>
997     <element name="Action" type="string" maxOccurs="unbounded"/>
998   </sequence>
999 </complexType>
1000
1001 <element name="Evidence" type="saml:AssertionSpecifierType"/>
1002
1003 <complexType name="AttributeAssertionType">
1004   <complexContent>
1005     <extension base="saml:SubjectAssertionType">
1006       <sequence>
1007         <element ref="saml:Attribute" maxOccurs="unbounded"/>
1008       </sequence>
1009     </extension>
1010   </complexContent>
1011 </complexType>
1012
1013 <element name="Attribute" type="saml:AttributeType"/>
1014 <complexType name="AttributeType">
1015   <sequence>
1016     <element name="AttributeName" type="string"/>
1017     <element name="AttributeNamespace"
1018       type="uriReference" minOccurs="0"/>
1019     <element name="AttributeValue" type="saml:AttributeValueType"
1020       minOccurs="0" maxOccurs="unbounded"/>
1021   </sequence>
1022 </complexType>
1023
1024 <complexType name="AttributeValueType">
1025   <sequence>
1026     <any namespace="##any" processContents="lax"
1027       minOccurs="0" maxOccurs="unbounded"/>
1028   </sequence>
1029 </complexType>
1030
1031 <element name="Conditions" type="saml:ConditionsType"/>
1032 <complexType name="ConditionsType">
1033   <sequence>
1034     <element name="Condition" type="saml:AbstractConditionType"
1035       minOccurs="0" maxOccurs="unbounded"/>
1036   </sequence>
1037   <attribute name="NotBefore" type="timeInstant" use="optional"/>
1038   <attribute name="NotOnOrAfter" type="timeInstant" use="optional"/>
1039 </complexType>
1040
1041 <complexType name="AbstractConditionType" abstract="true"/>
1042
1043 <complexType name="AudienceRestrictionConditionType">
1044   <complexContent>
1045     <extension base="saml:AbstractConditionType">
1046       <sequence>
1047         <element name="Audience" type="xsd:uriReference"
1048           minOccurs="0" maxOccurs="unbounded"/>
1049       </sequence>
1050     </extension>

```

```

1051     </complexContent>
1052 </complexType>
1053
1054 <element name="Advice" type="saml:AdviceType"/>
1055 <complexType name="AdviceType">
1056   <sequence>
1057     <any namespace="##any" processContents="lax"
1058       minOccurs="0" maxOccurs="unbounded"/>
1059   </sequence>
1060 </complexType>
1061 </schema>

```

5.2 Protocol Schema

```

1063 <?xml version="1.0" encoding="UTF-8"?>
1064 <schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/protocol/"
1065   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1066   xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
1067   xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"
1068   xmlns:samlp="http://www.oasis.org/tbs/1066-12-25/protocol/"
1069   xmlns="http://www.w3.org/2000/10/XMLSchema"
1070   elementFormDefault="unqualified">
1071   <import namespace="http://www.oasis.org/tbs/1066-12-25/"
1072     schemaLocation="draft-schema-assertion-10.xsd"/>
1073   <import namespace="http://www.w3.org/2000/09/xmldsig#"
1074     schemaLocation="xmldsig-core-schema.xsd"/>
1075   <annotation>
1076     <documentation>draft-schema-protocol-10.xsd</documentation>
1077 </annotation>
1078
1079   <simpleType name="CompletenessSpecifierType">
1080     <restriction base="string">
1081       <enumeration value="Any"/>
1082       <enumeration value="All"/>
1083     </restriction>
1084 </simpleType>
1085
1086   <simpleType name="StatusCodeType">
1087     <restriction base="string">
1088       <enumeration value="Success"/>
1089       <enumeration value="Failure"/>
1090       <enumeration value="Error"/>
1091       <enumeration value="Unknown"/>
1092     </restriction>
1093 </simpleType>
1094
1095   <complexType name="SAMLAbstractRequestType" abstract="true">
1096     <attribute name="RequestID" type="saml:IDType" use="required"/>
1097     <attribute name="Version" type="string" use="required"/>
1098 </complexType>
1099
1100   <element name="SAMLRequest" type="samlp:SAMLRequestType"/>
1101   <complexType name="SAMLRequestType">
1102     <complexContent>
1103       <extension base="samlp:SAMLAbstractRequestType">
1104         <choice>
1105           <element name="Query" type="samlp:SAMLQueryType"/>
1106           <element ref="saml:AssertionID" maxOccurs="unbounded"/>
1107         </choice>
1108       </extension>
1109     </complexContent>
1110 </complexType>
1111

```

```

1112 <complexType name="SAMLQueryType" abstract="true"/>
1113
1114 <complexType name="SubjectQueryType" abstract="true">
1115   <complexContent>
1116     <extension base="samlp:SAMLQueryType">
1117       <sequence>
1118         <element ref="saml:Subject"/>
1119       </sequence>
1120     </extension>
1121   </complexContent>
1122 </complexType>
1123
1124 <complexType name="AuthenticationQueryType">
1125   <complexContent>
1126     <extension base="samlp:SubjectQueryType">
1127       <sequence>
1128         <element ref="saml:AuthenticationCode" minOccurs="0"/>
1129         <!--do we want more than one of these?-->
1130       </sequence>
1131     </extension>
1132   </complexContent>
1133 </complexType>
1134
1135 <complexType name="AttributeQueryType">
1136   <complexContent>
1137     <extension base="samlp:SubjectQueryType">
1138       <sequence>
1139         <element ref="saml:Attribute"
1140           minOccurs="0" maxOccurs="unbounded"/>
1141         <element name="CompletenessSpecifier"
1142           type="samlp:CompletenessSpecifierType" default="All"/>
1143       </sequence>
1144     </extension>
1145   </complexContent>
1146 </complexType>
1147
1148 <element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>
1149 <complexType name="AuthorizationQueryType">
1150   <complexContent>
1151     <extension base="samlp:SubjectQueryType">
1152       <sequence>
1153         <element ref="saml:Evidence"
1154           minOccurs="0" maxOccurs="unbounded"/>
1155         <element ref="saml:Object"/>
1156       </sequence>
1157     </extension>
1158   </complexContent>
1159 </complexType>
1160
1161 <complexType name="SAMLAbstractResponseType" abstract="true">
1162   <attribute name="ResponseID" type="saml:IDType" use="required"/>
1163   <attribute name="InResponseTo" type="saml:IDType" use="required"/>
1164   <attribute name="Version" type="string" use="required"/>
1165 </complexType>
1166
1167 <element name="SAMLResponse" type="samlp:SAMLResponseType"/>
1168 <complexType name="SAMLResponseType">
1169   <complexContent>
1170     <extension base="samlp:SAMLAbstractResponseType">
1171       <sequence>
1172         <element ref="saml:Assertion"
1173           minOccurs="0" maxOccurs="unbounded"/>
1174       </sequence>

```

```
1175         <attribute name="StatusCode" type="samlp:StatusCodeType"  
1176                 use="required"/>  
1177         </extension>  
1178     </complexContent>  
1179 </complexType>  
1180  
1181 </schema>
```

Page: 1
[pm1]

Page: 5
[PHB2] Insert the class diagram here

Page: 5
[PHB3]This schema is actually to the previous version since that is what Spy 3.5 accepts.

Page: 5
[PHB4] We have to align with the OASIS convention here.

Page: 7
[PHB5]Here we need to redo the diagram to express the inheritance mechanism we choose in the end.

Page: 7
[PHB6] Need some better text here n'est pas?

Page: 10
[PHB7]Yikes! maybe we should put an IDType in here so that the security domain is at least unique!

Page: 10
[PHB8]But by whom? It may not have been the issuer of the assertion!

Page: 10
[PHB9] Where are these codes defined? Do we reuse the protocol elements of section 4, I think we should but in that case the schema should specify a URI.

Page: 14
[PHB10]But the current definition does not give a ground case, how about a string?

Page: 16
[PHB11]Add in here the substitution group issue.

Page: 18
[PHB12]Need to have text for these, how exactly does failure differ from error?

Page: 18
[PHB13] asking for it to be created.

Page: 19
[PHB14] asking for it to be created.

Page: 19
[PHB15] asking for it to be created.

Page: 22
[PHB16] 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
[PHB17]If we use substitution groups add in the text to specify the requirement