

1 **OASIS SSTC SAML Protocols Schema**
2 **Discussion**

3

4 draft-sstc-protocol-discussion-01.doc

5

6 28 July 2001

7 Authors:

8 Chris McLaren, Netegrity

9 Prateek Mishra, Netegrity

10

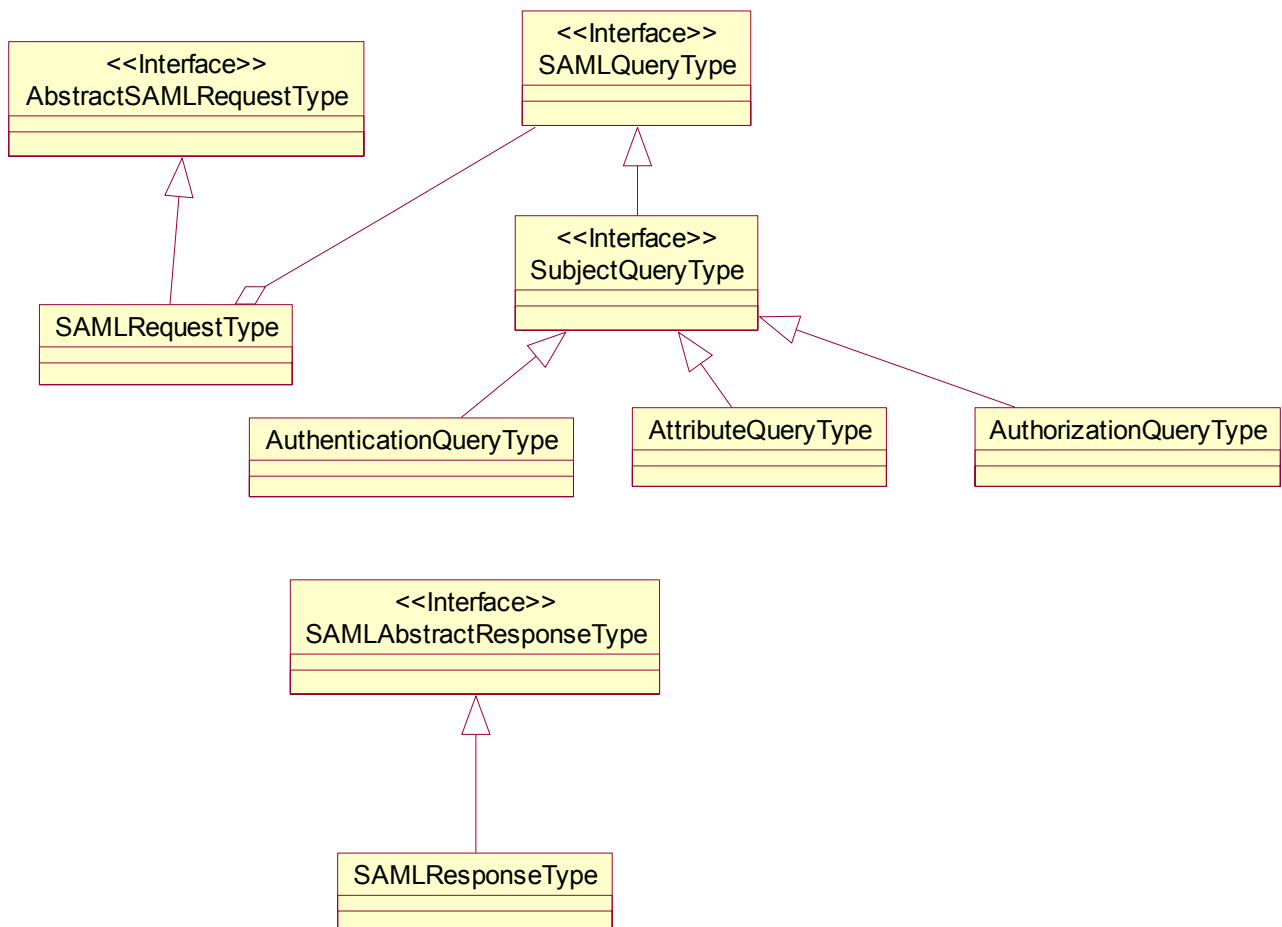
10		
11	<i>OASIS SSTC SAML Protocols Schema Discussion</i>	<i>1</i>
12	<i>1 Overview</i>	<i>3</i>
13	<i>2 Class Diagram</i>	<i>3</i>
14	<i>3 General Architecture</i>	<i>4</i>
15	<i>3.1 Discussion</i>	<i>4</i>
16	3.1.1 Inheritance Scheme	4
17	3.1.1.1 SAMLAbstractRequest and SAMLRequest	4
18	3.1.2 Example: SAMLRequest	5
19	3.1.3 SAMLQuery and SubjectQuery	5
20	3.1.3.1 Concrete Query Types	6
21	<i>3.2 Issues</i>	<i>6</i>
22	3.2.1 ISSUE:[PRO-01] Return of expired assertions?	6
23	3.2.2 ISSUE:[PRO-02] Assertion Lookup via Other Identifiers	6
24	<i>4 Authentication Query</i>	<i>6</i>
25	<i>4.1 Discussion</i>	<i>6</i>
26	4.1.1 Example: <SAMLQuery> with type AuthenticationQueryType	7
27	<i>5 Attribute Query</i>	<i>7</i>
28	<i>5.1 Discussion</i>	<i>7</i>
29	<i>5.2 Issues</i>	<i>8</i>
30	5.2.1 ISSUE:[PRO-03] ANY/ALL Qualifier	8
31	5.2.2 ISSUE:[PRO-04] Returning Attribute Names Only	8
32	5.2.3 ISSUE:[PRO-05] ALL or ``Error``	8
33	5.2.4 ISSUE:[PRO-06] Expressiveness of Query Language	9
34	<i>5.3 Example</i>	<i>9</i>
35	<i>6 Authorization Query</i>	<i>9</i>
36	<i>6.1 Discussion</i>	<i>9</i>
37	<i>6.2 Issues</i>	<i>10</i>
38	<i>6.3 Example</i>	<i>10</i>
39	<i>7 SAMLResponse and SAMLAbstractResponseType</i>	<i>11</i>
40	<i>7.1 Discussion</i>	<i>11</i>
41	7.1.1 SAMLAbstractResponseType	11
42	7.1.2 SAMLResponseType	11
43	<i>7.2 Issues</i>	<i>12</i>
44	7.2.1 ISSUE:[PRO-07 Status Code]	12
45	<i>7.3 Examples</i>	<i>12</i>
46	<i>8 References</i>	<i>13</i>
47		
48		

1 Overview

This document is the second document in a series of documents which provide a commentary on the SAML schema [draft-sstc-schema-assertion-12] and [draft-sstc-schema-protocols-12]. The first document [draft-sstc-core-discussion-01] describes the general principles governing the SAML schema design and comments on the assertion schema with examples.

2 Class Diagram

This section should contain a complete class diagram for draft-sstc-protocols-12. For now we have the following overview of a few key types types (interface represents an abstract type which cannot be directly instantiated):



79 3 General Architecture

80 3.1 Discussion

81 The SAML Protocols schema describes a request-response protocol for three different types of
 82 queries. Our architecture is based one request type, and one response type for all of these queries.
 83 The request type is capable of carrying many different types of queries all of which extend from
 84 one of two common abstract extension points. Three such concrete extensions are defined in the
 85 specification.

86 3.1.1 Inheritance Scheme

87 In order to ask a question in SAML one creates an instance of the SAMLRequest element. The
 88 SAMLRequest element's type definition is an extension of SAMLAbstractRequestType, which
 89 is provided as an extension point for potential future requirements.

90 One of possible content element in a SAMLRequest is a query. This query must be a descendant
 91 of the abstract SAMLQueryType. Three different such descendants are defined in the
 92 specification: AuthenticationQueryType, AttributeQueryType, and AuthorizationQueryType.
 93 Note that all of these are descended from the abstract base type SubjectQuery, which is in turn
 94 descended from the SAMLQueryType.

95 3.1.1.1 SAMLAbstractRequest and SAMLRequest

```

96 <xsd:complexType name="SAMLAbstractRequestType" abstract="true">
97   <xsd:attribute name="RequestID" type="saml:IDType" use="required"/>
98   <xsd:attribute name="Version" type="string" use="required"/>
99 </xsd:complexType>
100
101 <element name="SAMLRequestType" type="samlp:SAMLRequestType">
102 <complexType name="SAMLRequestType">
103   <complexContent>
104     <extension base="samlp:SAMLAbstractRequestType">
105       <choice>
106         <element ref="saml:SAMLQuery"/>
107         <element ref="saml:AssertionID"
108           maxOccurs="unbounded"/>
109       </choice>
110     </extension>
111   </complexContent>
112 </complexType>
  
```

113

114 The SAMLAbstractRequest defines only two attributes, both of which are required. It is an
 115 abstract type that can not be instantiated and is only provided as an extension point for possible
 116 future uses.

117 **RequestID:** This is a unique identifier for the request (See also ISSUE:[CONS-02])

118 **Version:** This is the string that identifies the version of the SAML protocol being used (See also
119 ISSUE:[CONS-05]).

120 The SAMLRequestType extends the abstract base by adding one of two elements to its body.
121 The element <SAMLRequest> is made available as a container for SAMLRequestType objects.

122 The element found within <SAMLRequest> indicates the type of the question asked:

123 If an <AssertionID> is present in the body the question is interpreted as “Would you return the
124 Assertion corresponding to this AssertionID?”

125 Otherwise some descendant of the abstract SAMLQueryType will be present, and will contain a
126 more specific question. These specific questions are detailed in §4 through §6.

127 *3.1.2 Example: SAMLRequest*

```
128 <SAMLRequest RequestID="urn:SAMLID:17862301" Version="100">
129   <AssertionID>urn:SAMLID:29100231XA</AssertionID>
130 </SAMLRequest>
```

131
132 In this example, the <SAMLRequest> element is requesting a SAML assertion with a certain
133 <AssertionID>. If successful, the assertion will be returned within a <SAMLResponse> element.

134 *3.1.3 SAMLQuery and SubjectQuery*

```
135 <complexType name="SAMLQueryType" abstract="true"/>
136
137 <complexType name="SubjectQueryType" abstract="true">
138   <complexContent>
139     <extension base="samlp:SAMLQueryType">
140       <sequence>
141         <element ref="saml:Subject"/>
142       </sequence>
143     </extension>
144   </complexContent>
145 </complexType>
```

146
147 The SAMLQuery element is an abstract element that exists to provide both an extension point
148 for possible future query types and could also act as the head element of a substitution group that
149 covers all possible query types. The type has no defined characteristics.

150 The SubjectQueryType is a second abstract type, derived from the SAMLQueryType with the
151 addition of one and only one <Subject> element. For information on the <Subject> element see
152 §4.1.3.1 of draft-sstc-core-XX-discussion.doc

153 The SubjectQueryType is the base class from which the three concrete query classes defined in
154 this specification are derived.

155 3.1.3.1 Concrete Query Types

156 Three concrete query types are defined in this specification: an authentication query, an attribute
 157 query, and an authorization query. Depending on what sort of question a SAMLRequest is
 158 contains, one of these types is used as the SAMLQuery descendant within the request. The query
 159 types defined in this specification are detailed in §4 through §6.

160 3.2 Issues

161 3.2.1 ISSUE:[PRO-01] Return of expired assertions?

162 Does the specification make any normative statements about the expiry state of assertions
 163 returned in response to SAMLRequests? Is it a requirement that only unexpired assertions are
 164 returned, or is the client responsible for checking? (*Seems pretty clear that the client will have to*
 165 *check anyway at time-of-use, so forcing the responder to check before replying seems like extra*
 166 *processing.*)

167 3.2.2 ISSUE:[PRO-02] Assertion Lookup via Other Identifiers

168 In some instances (such as the web browser profile) it is necessary to lookup an assertion using
 169 an identifier other than the <AssertionID>. Typically, such an identifier is opaque and may have
 170 been created in some proprietary way by an asserting party. Do we need an additional element in
 171 SAMLRequestType to model this type of lookup?

172 4 Authentication Query

173 4.1 Discussion

```
174 <complexType name="AuthenticationQueryType">
175   <complexContent>
176     <extension base="samlp:SubjectQueryType">
177       <sequence>
178         <element ref="saml:AuthenticationCode"
179           minOccurs="0"/>
180       </sequence>
181     </extension>
182   </complexContent>
183 </complexType>
```

184
 185 The AuthenticationQueryType asks the question “What authentication assertions are available
 186 for this Subject?”

187 An AuthenticationQuery contains all the elements and attributes of a SubjectQuery and extends
 188 them with the optional addition of an AuthenticationCode. The <AuthenticationCode> element is
 189 defined in §4.1.4 of draft-sstc-core-XX-discussion.doc. In the context of a query this code can be

190 used to as a filter for possible responses. This supports the query “What authentication assertions
191 do you have for this Subject with the following AuthenticationCode?”

192 A SAML processor will return a certain number of AuthenticationAssertions in response to a
193 SAMLQuery of type AuthenticationQueryType. The <Subject> and <AuthenticationCode> of
194 the returned assertions must be identical to the subject (and optional <AuthenticationCode>)
195 fields of the SAMLQuery. There is no implication that all such assertions must be returned.

196 **4.1.1 Example: <SAMLQuery> with type AuthenticationQueryType**

```
197 <SAMLQuery xsi:type="samlp:AuthenticationQueryType" >
198   <Subject>
199     <NameIdentifier>
200       <SecurityDomain>www.example.com</SecurityDomain>
201       <Name>cn=SomeUser,co=example,ou=sales</Name>
202     </NameIdentifier>
203   </Subject>
204   <AuthenticationCode>X509v3</AuthenticationCode>
205 </SAMLQuery>
```

206

207 In this example we are asking for all authentication assertions for a subject with “cn=SomeUser,
208 co=example, ou=sales” with <AuthenticationCode> set to X509v3.

209 When wrapped in a <SAMLRequest> element we have the complete query:

```
210 <SAMLRequest RequestID="{EE52CAF4-3768-4ebe-84D3-4D372C892A5D}"
211   Version="http://www.oasis.org/tbs/1066-12-25/protocol/" >
212   <SAMLQuery xsi:type="samlp:AuthenticationQueryType" >
213     <Subject>
214       <NameIdentifier>
215         <SecurityDomain>www.example.com</SecurityDomain>
216         <Name>SomeUser</Name>
217       </NameIdentifier>
218     </Subject>
219     <AuthenticationCode>X509v3</AuthenticationCode>
220   </SAMLQuery>
221 </SAMLRequest>
```

222

223 **5 Attribute Query**

224 **5.1 Discussion**

```
225 <complexType name="AttributeQueryType">
226   <complexContent>
227     <extension base="samlp:SubjectQueryType">
228       <sequence>
229         <element ref="saml:Attribute" minOccurs="0"
230   maxOccurs="unbounded"/>
231         <element name="CompletenessSpecifier"
232   type="samlp:CompletenessSpecifierType"
```

```
233         default="All"/>
234     </sequence>
235 </extension>
236 </complexContent>
237 </complexType>
```

238

239 The AttributeQueryType can be used to ask the questions:

- 240 • “Return all of the attributes for this Subject (that I am allowed to see)?” The response
241 will be in the form of an Attribute assertion.
- 242 • “Return ALL of the following attributes for this Subject?” The response will be in the
243 form an Attribute assertion. If the requestor is not allowed access to ALL the requested
244 attributes he will get NONE of them.
- 245 • “Return me ANY of the following attributes for this Subject?” The response will be in
246 the form an Attribute assertion. Any attributes which the requestor is not allowed access
247 to will not be present in the returned assertion.

248 An AttributeQuery contains all the elements and attributes of a SubjectQuery and extends them
249 with the addition of zero or more Attributes and exactly one CompletenessSpecifier.

250 The set of Attributes are used to specify the list of attributes in the query in the second and third
251 forms of the question. If there are no Attributes provided the query is interpreted as being the
252 first question above and the CompletenessSpecifier is ignored.

253 The CompletenessSpecifier is a two-value enumeration of “Any” or “All”. The chosen value
254 selects between the second and third questions listed above.

255 **5.2 Issues**

256 ***5.2.1 ISSUE:[PRO-03] ANY/ALL Qualifier***

257 Are these qualifiers necessary? Can we remove them?

258 See also ISSUE:[F2F#3-20, 3-21].

259 ***5.2.2 ISSUE:[PRO-04] Returning Attribute Names Only***

260 Do we need an additional query format where only the attribute names are returned? For
261 example, the query where all the attribute names for a subject are returned.

262 See also ISSUE:[F2F#3-22, 23].

263 ***5.2.3 ISSUE:[PRO-05] ALL or ``Error``***

264 Is the all or “error” semantics for the ALL qualifier appropriate? Should we just follow LDAP
265 semantics for this type of query?

266 See also [ISSUE:F2F#3-24] and discussion [F3] from [f2f#3-minutes]

267 **5.2.4 ISSUE:[PRO-06] Expressiveness of Query Language**

268 The approach taken in this draft is to use a very simple query language for attributes, essentially
 269 consisting of only of attribute names. At the f2f#3, there were suggestions that a richer query
 270 language might be used, such as for example based on XPATH. Is there any need to further
 271 explore this direction for SAML 1.0?

272 See also [F2] in [f2f#3-minutes]

273 **5.3 Example**

```

274 <SAMLQuery xsi:type="samlp:AttributeQueryType">
275   <Subject>
276     <NameIdentifier>
277       <SecurityDomain>www.example.com</SecurityDomain>
278       <Name>SomeUser</Name>
279     </NameIdentifier>
280   </Subject>
281   <CompletenessSpecifier>ANY</CompletenessSpecifier>
282   <Attribute>
283     <AttributeName>NetWorthSummary</AttributeName>
284     <AttributeNamespace>
285       http://ns.finance-vocab.org/finance
286     </AttributeNamespace>
287   </Attribute>
288   <Attribute>
289     <AttributeName>CreditScore</AttributeName>
290     <AttributeNamespace>
291       http://ns.finance-vocab.org/finance
292     </AttributeNamespace>
293   </Attribute>
294 </SAMLQuery>
  
```

295

296 This example is requesting two namespace-qualified attributes for the given Subject: a
 297 “NetWorthSummary” and a “CreditScore”. Since it is requested with the ANY flag that means
 298 the response will contain values for zero, one or both of these attribute names.

299 **6 Authorization Query**

300 **6.1 Discussion**

```

301 <complexType name="AuthorizationQueryType">
302   <complexContent>
303     <extension base="samlp:SubjectQueryType">
304       <sequence>
305         <element ref="saml:Evidence" minOccurs="0"
306           maxOccurs="unbounded"/>
307         <element ref="saml:Object"/>
308       </sequence>
309     </extension>
  
```

```

310     </complexContent>
311 </complexType>

```

312

313 An Authorization Query is used to ask only one type of question: “Should action(s) Y on
 314 resource Z be allowed for subject S given evidence E?” The answer comes in the form of an
 315 Authorization Decision assertion. The action(s) and resource are optionally namespace-scoped
 316 and are contained with in the <Object> element of the query.

317 An AuthorizationQuery contains all the elements and attributes of a SubjectQuery and extends
 318 them with the addition of exactly one Object element and a collection zero or more assertions or
 319 assertionIDs to be used as evidence.

320 The Evidence and Object elements are defined in draft-sstc-core-XX-discussion.doc; Evidence in
 321 §4.1.6 and Object in §4.1.6.1.

322 6.2 Issues

323 6.3 Example

```

324 <SAMLQuery xsi:type="samlp:AuthorizationQueryType">
325   <Subject>
326     <NameIdentifier>
327       <SecurityDomain>us-staff</SecurityDomain>
328       <Name>cn=SomeUser,ou=finance,co=example</Name>
329     </NameIdentifier>
330   </Subject>
331   <Object>
332     <Resource>
333       http://www.example.com/confidential/agree.html
334     </Resource>
335     <Action>GET</Action>
336     <Action>POST</Action>
337     <Namespace>urn:samlaction:HTTP</Namespace>
338   </Object>
339   <Evidence>
340     <AssertionID>{EE52CAF4-3452-4ebe-84D3-4D372C892A5D}</AssertionID>
341   </Evidence>
342 </SAMLQuery>

```

343

344 In this example, the query concerns whether subject “cn=SomeUser,...” has the right to GET
 345 AND POST on a certain URL. The actions GET, POST are taken from the namespace
 346 urn:samlaction:HTTP.

347 7 SAMLResponse and 348 SAMLAbstractResponseType

349 7.1 Discussion

```

350 <complexType name="SAMLAbstractResponseType" abstract="true">
351   <attribute name="ResponseID" type="saml:IDType" use="required"/>
352   <attribute name="InResponseTo" type="saml:IDType" use="required"/>
353   <attribute name="Version" type="string" use="required"/>
354 </complexType>
355 <element name="SAMLResponse" type="SAMLResponseType"/>
356 <complexType name="SAMLResponseType">
357   <complexContent>
358     <extension base="samlp:SAMLAbstractResponseType">
359       <sequence>
360         <element ref="saml:Assertion" minOccurs="0"
361           maxOccurs="unbounded"/>
362       </sequence>
363       <attribute name="StatusCode" type="samlp:StatusCodeType"
364         use="required"/>
365     </extension>
366   </complexContent>
367 </complexType>

```

368

369 All types of requests in SAML are met with a common response, the <SAMLResponse>, which
370 has type SAMLResponseType. SAMLResponseType is an extension of the abstract base type
371 SAMLAbstractResponseType, which is provided as an extension point for other possible future
372 needs with respect to responses.

373 7.1.1 SAMLAbstractResponseType

374 The base type defines only the three required attributes of a response: the ResponseID, the
375 InResponseTo identifier, and the version string.

376 **RequestID:** This is the identifier for the response. See also [ISSUE:CONS-02]

377 **InResponseTo:** This attribute holds the identifier of the request which this response is
378 answering. Naturally whatever formatting rules, if any, that are arrived at for the request
379 identifiers will also apply to the type of this attribute.

380 **Version:** This is the string that identifies the version of the SAML protocol being used by the
381 responder. See also [ISSUE:CONS-05].

382 7.1.2 SAMLResponseType

383 The SAMLResponseType extends the SAMLAbstractResponseType with the addition of a
384 required Status Code attribute and a series of zero or more assertions that represent the answers
385 to the original request.

386 **StatusCode:** This attribute holds a string that describes the result of the original request, as
 387 expressed in this response. The contents of this attribute are restricted to members of an
 388 enumeration defined in the type “StatusCodeType” (see below).

```
389 <simpleType name="StatusCodeType">
390   <restriction base="string">
391     <enumeration value="Success"/>
392     <enumeration value="Failure"/>
393     <enumeration value="Error"/>
394     <enumeration value="Unknown"/>
395   </restriction>
396 </simpleType>
```

397
 398 The assertions carried in the response represent the answers to the original question. The
 399 responding entity is responsible for what assertions are provided in response to any question.

400 7.2 Issues

401 7.2.1 ISSUE:[PRO-07 Status Code]

402 Are the status codes listed for StatusCodeType sufficient? If not how do we want to define a
 403 bigger list: keep it open with well-known values, use someone else’s list, define an extension
 404 system, etc.

405 See also ISSUE:[F2F#3-33, 34].

406 7.3 Examples

407 The example below shows an example of a successful response to the AttributeQuery shown in
 408 Section 5.3.

```
409 <SAMLResponse ResponseID="{EF52CBF4-3452-4ebe-84D3-4D372C892A5D}"
410   InResponseTo= "{EE52CAF4-3768-4ebe-84D3-4D372C892A5D}"
411   Version="0100"
412   StatusCode="Success">
413   <Assertion xsi:type="saml:AttributeAssertionType"
414     version="http://www.oasis.org/tbs/1066-12-25/"
415     AssertionID="{EE52CAF4-3452-4ebe-84D3-4D372C892A5D}"
416     Issuer="www.example.com"
417     IssueInstant="2001-05-31T13:20:00-05:00">
418     <Conditions NotBefore="2001-05-31T13:20:00-05:00"
419       NotOnOrAfter="2001-05-31T13:25:00-05:00" />
420     <Subject>
421       <NameIdentifier>
422         <SecurityDomain>www.example.com</SecurityDomain>
423         <Name>SomeUser</Name>
424       </NameIdentifier>
425     </Subject>
426     <Attribute>
427       <AttributeName>NetWorthSummary</AttributeName>
428       <AttributeNameSpace>
```

```

429         http://ns.finance-vocab.org/finance
430     </AttributeNameSpace>
431     <AttributeValue>
432         <CreditSummary>
433             <HistoryScore>Excellent</HistoryScore>
434             <CurrentAssets>Loaded</CurrentAsserts>
435         </CreditSummary>
436     </AttributeValue>
437 </Attribute>
438 </Assertion>
439 </SAMLResponse>

```

440

441 The example below shows an example of a successful response to the AuthorizationQuery
442 shown in Section 6.3.

443

```

444 <SAMLResponse ResponseID="{EF52CBF4-3452-4ebe-84D3-4D372C892A5D}"
445   InResponseTo= "{EE52CAF4-3768-4ebe-84D3-4D372C892A5D}"
446   Version="0100"
447   StatusCode="Success">
448     <Assertion xsi:type="saml:AuthorizationDecisionAssertionType"
449       version="http://www.oasis.org/tbs/1066-12-25/"
450       AssertionID="{EE52CAF4-3452-4ebe-84D3-4D372C892A5D}"
451       Issuer="www.example.com"
452       IssueInstant="2001-05-31T13:20:00-05:00">
453       <Conditions NotBefore="2001-05-31T13:20:00-05:00"
454         NotOnOrAfter="2001-05-31T13:25:00-05:00" />
455     <Subject>
456       <NameIdentifier>
457         <SecurityDomain>us-staff</SecurityDomain>
458         <Name>cn=SomeUser,ou=finance,co=example</Name>
459       </NameIdentifier>
460     </Subject>
461     <Object>
462       <Resource>
463         http://www.example.com/confidential/agree.html
464       </Resource>
465       <Action>GET</Action>
466       <Action>POST</Action>
467       <Namespace>urn:samlaction:HTTP</Namespace>
468     </Object>
469     <Evidence>
470       <AssertionID>{EE52CAF4-3452-4ebe-84D3-4D372C892A5D}</AssertionID>
471     </Evidence>
472     <Answer>Permit</Answer>
473   </Assertion>
474 </SAMLResponse>

```

475 8 References

476 [f2f#3-minutes] [http://www.oasis-open.org/committees/security/minutes/SSTC-F2F-3-Minutes-](http://www.oasis-open.org/committees/security/minutes/SSTC-F2F-3-Minutes-01.txt)
477 [01.txt](http://www.oasis-open.org/committees/security/minutes/SSTC-F2F-3-Minutes-01.txt)

- 478 [draft-sstc-core-discussion-01] [http://www.oasis-open.org/committees/security/docs/draft-sstc-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-discussion-01.pdf)
479 [core-discussion-01.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-discussion-01.pdf)
- 480 [draft-sstc-schema-assertion-12] [http://www.oasis-open.org/committees/security/docs/draft-sstc-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-12.xsd)
481 [schema-assertion-12.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-12.xsd)
- 482 [draft-sstc-schema-protocols-12] [http://www.oasis-open.org/committees/security/docs/draft-sstc-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-12.xsd)
483 [schema-protocol-12.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-12.xsd)