



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

**Document identifier:** draft-sstc-core-22

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

**Publication date:** 31st December 2001

**Status:** Interim draft. Send comments to the editors.

**Editors:**

Phillip Hallam-Baker, VeriSign, ([pbaker@verisign.com](mailto:pbaker@verisign.com))

Eve Maler, Sun Microsystems, ([eve.maler@sun.com](mailto:eve.maler@sun.com))

**Contributors:**

Carlisle Adams, Entrust

Marc Chanliau, Netegrity

Nigel Edwards, Hewlett-Packard

Marlena Erdos, Tivoli

Simon Godik, Crosslogic

Jeff Hodges, Oblix

Charles Knouse, Oblix

Chris McLaren, Netegrity

Prateek Mishra, Netegrity

RL "Bob" Morgan, University of Washington

Tim Moses, Entrust

David Orchard, BEA

Joe Pato, Hewlett Packard

Darren Platt, RSA

Irving Reid, Baltimore

28		
29	ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP	
30	LANGUAGE (SAML)	1
31	<b>1. INTRODUCTION</b>	<b>6</b>
32	1.1. NOTATION	6
33	1.2. SCHEMA ORGANIZATION AND NAMESPACES	6
34	1.3. SAML CONCEPTS (NON-NORMATIVE)	7
35	<b>2. SAML ASSERTIONS</b>	<b>8</b>
36	2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	8
37	2.2. SIMPLE TYPES	8
38	2.2.1. <i>Simple Type IDType</i>	8
39	2.2.2. <i>Simple Type DecisionType</i>	9
40	2.3. ASSERTIONS	9
41	2.3.1. <i>Element &lt;AssertionSpecifier&gt;</i>	9
42	2.3.2. <i>Element &lt;AssertionID&gt;</i>	10
43	2.3.3. <i>Element &lt;Assertion&gt;</i>	10
44	2.3.3.1. <i>Element &lt;Conditions&gt;</i>	11
45	2.3.3.1.1. <i>Attributes NotBefore and NotOnOrAfter</i>	12
46	2.3.3.1.2. <i>Element &lt;Condition&gt;</i>	12
47	2.3.3.1.3. <i>Elements &lt;AudienceRestrictionCondition&gt; and &lt;Audience&gt;</i>	12
48	2.3.3.1.4. <i>Condition Type TargetRestrictionType</i>	13
49	2.3.3.2. <i>Elements &lt;Advice&gt; and &lt;AdviceElement&gt;</i>	13
50	2.4. STATEMENTS	14
51	2.4.1. <i>Element &lt;Statement&gt;</i>	14
52	2.4.2. <i>Element &lt;SubjectStatement&gt;</i>	14
53	2.4.2.1. <i>Element &lt;Subject&gt;</i>	14
54	2.4.2.2. <i>Element &lt;NameIdentifier&gt;</i>	15
55	2.4.2.3. <i>Elements &lt;SubjectConfirmation&gt;, &lt;ConfirmationMethod&gt;, and &lt;SubjectConfirmationData&gt;</i>	15
56	2.4.3. <i>Element &lt;AuthenticationStatement&gt;</i>	16

57	2.4.3.1. Element <AuthenticationLocality>	16
58	<b>2.4.4. Element &lt;AuthorizationDecisionStatement&gt;</b>	17
59	2.4.4.1. Elements <Actions> and <Action>	17
60	2.4.4.2. Element <Evidence>	18
61	<b>2.4.5. Element &lt;AttributeStatement&gt;</b>	18
62	2.4.5.1. Elements <AttributeDesignator> and <Attribute>	18
63	2.4.5.1.1 Element <AttributeValue>	19
64	<b>3. SAML PROTOCOL</b>	<b>20</b>
65	<b>3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</b>	20
66	<b>3.2. SIMPLE TYPES</b>	20
67	<b>3.2.1. Simple Type StatusCodeType</b>	20
68	<b>3.3. REQUESTS</b>	21
69	<b>3.3.1. Complex Type RequestAbstractType</b>	21
70	3.3.1.1. Element <RespondWith>	21
71	<b>3.3.2. Element &lt;Request&gt;</b>	22
72	<b>3.4. QUERIES</b>	23
73	<b>3.4.1. Element &lt;Query&gt;</b>	23
74	<b>3.4.2. Element &lt;SubjectQuery&gt;</b>	23
75	<b>3.4.3. Element &lt;AuthenticationQuery&gt;</b>	23
76	<b>3.4.4. Element &lt;AttributeQuery&gt;</b>	24
77	<b>3.4.5. Element &lt;AuthorizationDecisionQuery&gt;</b>	24
78	<b>3.5. RESPONSES</b>	25
79	<b>3.5.1. Complex Type ResponseAbstractType</b>	25
80	<b>3.5.2. Element &lt;Response&gt;</b>	25
81	3.5.2.1. Element <StatusReason>	26
82	<b>4. SAML VERSIONING</b>	<b>27</b>
83	<b>4.1. ASSERTION VERSION</b>	27
84	<b>4.2. REQUEST VERSION</b>	27

85	4.3. RESPONSE VERSION	28
86	<b>5. SAML EXTENSIONS</b>	<b>29</b>
87	5.1. ASSERTION SCHEMA EXTENSION	29
88	5.2. PROTOCOL SCHEMA EXTENSION	29
89	5.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	30
90	<b>6. SAML-DEFINED IDENTIFIERS</b>	<b>31</b>
91	6.1. CONFIRMATION METHOD IDENTIFIERS	31
92	6.1.1. SAML Artifact:	31
93	6.1.2. SAML Artifact (SHA-1):	31
94	6.1.3. Holder of Key:	31
95	6.1.4. Sender Vouches:	31
96	6.1.5. Password (Pass-Through):	31
97	6.1.6. Password (One-Way-Function SHA-1):	32
98	6.1.7. Kerberos [Kerberos]	32
99	6.1.8. SSL/TLS Certificate Based Client Authentication:	32
100	6.1.9. Object Authenticator (SHA-1):	32
101	6.1.10. PKCS#7	32
102	6.1.11. Cryptographic Message Syntax	33
103	6.1.12. XML Digital Signature	33
104	6.2. ACTION NAMESPACE IDENTIFIERS	33
105	6.2.1. Read/Write/Execute/Delete/Control:	33
106	6.2.2. Read/Write/Execute/Delete/Control with Negation:	33
107	6.2.3. Get/Head/Put/Post:	34
108	6.2.4. UNIX File Permissions:	34
109	<b>7. SAML SCHEMA LISTINGS</b>	<b>35</b>

110	<b>7.1.</b> ASSERTION SCHEMA	35
111	<b>7.2.</b> PROTOCOL SCHEMA	38
112	<b>8.</b> REFERENCES	41
113	<b>APPENDIX A. NOTICES</b>	43
114		

# 1. Introduction

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

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

## 1.1. Notation

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

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

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

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

Listings of SAML schemas appear like this.

Example code listings appear like this.

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

?? The prefix `saml:` stands for the SAML assertion namespace.

?? The prefix `samlp:` stands for the SAML request-response protocol namespace.

?? The prefix `ds:` stands for the W3C XML Signature namespace.

?? The prefix `xsd:` stands for the W3C XML Schema namespace in example listings. In schema listings, this is the default namespace and no prefix is shown.

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

## 1.2. Schema Organization and Namespaces

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

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

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

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

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

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

161       <http://www.w3.org/2000/09/xmldsig#>

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

## 164       **1.3. SAML Concepts (Non-Normative)**

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

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

## 2. SAML Assertions

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

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

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

### 2.1. Schema Header and Namespace Declarations

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

```
<schema
  targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-22.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-22.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified">
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd"/>
  <annotation>
    <documentation>draft-sstc-schema-assertion-22.xsd</documentation>
  </annotation>
  ...
</schema>
```

### 2.2. Simple Types

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

#### 2.2.1. Simple Type IDType

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

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

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

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



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

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

```
218 <simpleType name="IDType">  
219   <restriction base="string"/>  
220 </simpleType>
```

## 221 2.2.2. Simple Type DecisionType

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

224 Permit

225       The specified action is permitted.

226 Deny

227       The specified action is denied.

228 Indeterminate

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

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

```
231 <simpleType name="DecisionType">  
232   <restriction base="string">  
233     <enumeration value="Permit"/>  
234     <enumeration value="Deny"/>  
235     <enumeration value="Indeterminate"/>  
236   </restriction>  
237 </simpleType>
```

## 238 2.3. Assertions

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

### 240 2.3.1. Element <AssertionSpecifier>

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

243 <AssertionID>

244       Specifies an assertion by reference to the value of the assertion's AssertionID  
245       attribute.

246 <Assertion>

247       Specifies an assertion by value.

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

249 **AssertionSpecifierType** complex type:

```
250 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>  
251 <complexType name="AssertionSpecifierType">  
252   <choice>  
253     <element ref="saml:AssertionID"/>  
254     <element ref="saml:Assertion"/>  
255   </choice>  
256 </complexType>
```

### 2.3.2. Element <AssertionID>

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

The following schema fragment defines the <AssertionID> element:

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

### 2.3.3. Element <Assertion>

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

**MajorVersion** [Required]

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

**MinorVersion** [Required]

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

**AssertionID** [Required]

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

**Issuer** [Required]

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

**IssueInstant** [Required]

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

**<Conditions>** [Optional]

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

**<Advice>** [Optional]

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

One or more of the following statement elements:

**<Statement>**

A statement defined in an extension schema.

**<SubjectStatement>**

A subject statement defined in an extension schema.

**<AuthenticationStatement>**

An authentication statement.

**<AuthorizationDecisionStatement>**

An authorization decision statement.

**<AttributeStatement>**

An attribute statement.

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

```
<complexType name="AssertionType">
```

```

301     <sequence>
302       <element ref="saml:Conditions" minOccurs="0"/>
303       <element ref="saml:Advice" minOccurs="0"/>
304       <choice minOccurs="0" maxOccurs="unbounded">
305         <element ref="saml:Statement"/>
306         <element ref="saml:SubjectStatement"/>
307         <element ref="saml:AuthenticationStatement"/>
308         <element ref="saml:AuthorizationDecisionStatement"/>
309         <element ref="saml:AttributeStatement"/>
310       </choice>
311     </sequence>
312     <attribute name="MajorVersion" type="integer" use="required"/>
313     <attribute name="MinorVersion" type="integer" use="required"/>
314     <attribute name="AssertionID" type="saml:IDType" use="required"/>
315     <attribute name="Issuer" type="string" use="required"/>
316     <attribute name="IssueInstant" type="dateTime" use="required"/>
317   </complexType>

```

### 318 2.3.3.1. Element <Conditions>

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

- 323 ?? If any condition evaluates to Invalid, the assertion status is Invalid.
- 324 ?? If no condition evaluates to Invalid and one or more conditions evaluate to  
325 Indeterminate, the assertion status is Indeterminate.
- 326 ?? If no conditions are supplied or all the specified conditions evaluate to Valid, the  
327 assertion status is Valid.

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

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

332 NotBefore [Optional]

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

334 NotOnOrAfter [Optional]

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

336 <Condition> [Zero or more]

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

338 <AudienceRestrictionCondition> [Any Number]

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

340 <TargetRestrictionCondition> [Any Number]

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

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

```

345   <element name="Conditions" type="saml:ConditionsType"/>
346   <complexType name="ConditionsType">
347     <choice minOccurs="0" maxOccurs="unbounded">
348       <element ref="saml:Condition"/>
349       <element ref="saml:AudienceRestrictionCondition"/>

```

```

350     <element ref="saml:TargetRestrictionCondition"/>
351   </choice>
352   <attribute name="NotBefore" type="dateTime" use="optional"/>
353   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
354 </complexType>

```

#### 355 2.3.3.1.1 *Attributes NotBefore and NotOnOrAfter*

356 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion.

357 The `NotBefore` attribute specifies the time instant at which the validity interval begins. The

358 `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.

359 If the value for either `NotBefore` or `NotOnOrAfter` is omitted or is equal to the start of the  
360 epoch, it is considered unspecified. If the `NotBefore` attribute is unspecified (and if any other  
361 conditions that are supplied evaluate to `Valid`), the assertion is valid at any time before the time  
362 instant specified by the `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified  
363 (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid from the  
364 time instant specified by the `NotBefore` attribute with no expiry. If neither attribute is specified  
365 (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid at any  
366 time.

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

#### 371 2.3.3.1.2 *Element <Condition>*

372 The `<Condition>` element serves as an extension point for new conditions. Its

373 **ConditionAbstractType** complex type is abstract; extension elements MUST use the `xsi:type`  
374 attribute to indicate the derived type.

375 The following schema fragment defines the `<Condition>` element and its

376 **ConditionAbstractType** complex type:

```

377   <element name="Condition" type="saml:ConditionAbstractType"/>
378   <complexType name="ConditionAbstractType" abstract="true"/>

```

#### 379 2.3.3.1.3 *Elements <AudienceRestrictionCondition> and <Audience>*

380 The `<AudienceRestrictionCondition>` element specifies that the assertion is addressed to  
381 one or more specific audiences. Although a party that is outside the audiences specified is  
382 capable of drawing conclusions from an assertion, the issuer explicitly makes no representation  
383 as to accuracy or trustworthiness to such a party.

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

386 The condition evaluates to `Valid` if and only if the relying party is a member of one or more of  
387 the audiences specified.

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

394 The following schema fragment defines the `<AudienceRestrictionCondition>` element and  
395 its **AudienceRestrictionConditionType** complex type:

```

396     <element name="AudienceRestrictionCondition"
397           type="saml:AudienceRestrictionConditionType" />
398   <complexType name="AudienceRestrictionConditionType">
399     <complexContent>
400       <extension base="saml:ConditionAbstractType">
401         <sequence>
402           <element ref="saml:Audience"
403             minOccurs="1" maxOccurs="unbounded" />
404         </sequence>
405       </extension>
406     </complexContent>
407   </complexType>
408   <element name="Audience" type="anyURI" />

```

#### 409 2.3.3.1.4 Condition Type TargetRestrictionType

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

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

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

```

416   <element name="TargetRestrictionCondition"
417         type="saml:TargetRestrictionConditionType" />
418   <complexType name="TargetRestrictionConditionType">
419     <complexContent>
420       <extension base="saml:ConditionAbstractType">
421         <sequence>
422           <element ref="saml:Target"
423             minOccurs="1" maxOccurs="unbounded" />
424         </sequence>
425       </extension>
426     </complexContent>
427   </complexType>
428   <element name="Target" type="anyURI" />

```

#### 429 2.3.3.2. Elements <Advice> and <AdviceElement>

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

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

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

- 437 ?? Include evidence supporting the assertion claims to be cited, either directly (through  
 438 incorporating the claims) or indirectly (by reference to the supporting assertions).
- 439 ?? State a proof of the assertion claims.
- 440 ?? Specify the timing and distribution points for updates to the assertion.

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

```

443   <element name="Advice" type="saml:AdviceType" />
444   <complexType name="AdviceType">
445     <sequence>

```

```

446         <choice minOccurs="0" maxOccurs="unbounded">
447             <element ref="saml:AssertionSpecifier"/>
448             <element ref="saml:AdviceElement"/>
449             <any namespace="##other" processContents="lax"/>
450         </choice>
451     </sequence>
452 </complexType>
453 <element name="AdviceElement" type="saml:AdviceAbstractType"/>
454 <complexType name="AdviceAbstractType"/>

```

## 2.4. Statements

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

### 2.4.1. Element <Statement>

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

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

```

463 <element name="Statement" type="saml:StatementAbstractType"/>
464 <complexType name="StatementAbstractType" abstract="true"/>

```

### 2.4.2. Element <SubjectStatement>

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

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

```

473 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
474 <complexType name="SubjectStatementAbstractType" abstract="true">
475     <complexContent>
476         <extension base="saml:StatementAbstractType">
477             <sequence>
478                 <element ref="saml:Subject"/>
479             </sequence>
480         </extension>
481     </complexContent>
482 </complexType>

```

#### 2.4.2.1. Element <Subject>

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

<NameIdentifier>

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

<SubjectConfirmation>

Information that allows the subject to be authenticated.

If a <Subject> element contains more than one subject specification, the issuer is asserting that the surrounding statement is true for all of the subjects specified. For example, if both a <NameIdentifier> and a <SubjectConfirmation> element are present, the issuer is

asserting that the statement is true of both subjects being identified. A <Subject> element SHOULD NOT identify more than one principal.

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

```
<element name="Subject" type="saml:SubjectType"/>
<complexType name="SubjectType">
  <choice maxOccurs="unbounded">
    <sequence>
      <element ref="saml:NameIdentifier"/>
      <element ref="saml:SubjectConfirmation" minOccurs="0"/>
    </sequence>
    <element ref="saml:SubjectConfirmation"/>
  </choice>
</complexType>
```

#### 2.4.2.2. Element <NameIdentifier>

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

SecurityDomain

The security domain governing the name of the subject.

Name

The name of the subject.

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

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

```
<element name="NameIdentifier" type="saml:NameIdentifierType"/>
<complexType name="NameIdentifierType">
  <attribute name="SecurityDomain" type="string"/>
  <attribute name="Name" type="string"/>
</complexType>
```

#### 2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>

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

<ConfirmationMethod> [One or more]

A URI that identifies a protocol to be used to authenticate the subject. URIs identifying common authentication protocols are listed in Section 6.

<SubjectConfirmationData> [Zero or more]

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

<ds:KeyInfo> [Optional]

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

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

```
<element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
<complexType name="SubjectConfirmationType">
```



```

541     <sequence>
542       <element ref="saml:ConfirmationMethod" maxOccurs="unbounded" />
543       <element ref="saml:SubjectConfirmationData" minOccurs="0" />
544       <element ref="ds:KeyInfo" minOccurs="0" />
545     </sequence>
546   </complexType>
547   <element name="SubjectConfirmationData" type="string" minOccurs="0" />
548   <element name="ConfirmationMethod" type="anyURI" />

```

### 2.4.3. Element <AuthenticationStatement>

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

**AuthenticationMethod** [Required]

A URI that specifies the type of authentication that took place. URIs identifying common authentication protocols are listed in Section 6.

**AuthenticationInstant** [Required]

Specifies the time at which the authentication took place.

**<AuthenticationLocality>** [Optional]

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

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

```

564   <element name="AuthenticationStatement"
565         type="saml:AuthenticationStatementType" />
566   <complexType name="AuthenticationStatementType">
567     <complexContent>
568       <extension base="saml:SubjectStatementAbstractType">
569         <sequence>
570           <element ref="saml:AuthenticationLocality" minOccurs="0" />
571         </sequence>
572         <attribute name="AuthenticationMethod" type="anyURI" />
573         <attribute name="AuthenticationInstant" type="dateTime" />
574       </extension>
575     </complexContent>
576   </complexType>

```

#### 2.4.3.1. Element <AuthenticationLocality>

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

**IPAddress** [Optional]

The IP address of the system entity that was authenticated.

**DNSAddress** [Required]

The DNS address of the system entity that was authenticated.

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

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

```

588   <element name="AuthenticationLocality"
589         type="saml:AuthenticationLocalityType" />
590   <complexType name="AuthenticationLocalityType">

```



```

591     <attribute name="IPAddress" type="string" use="optional" />
592     <attribute name="DNSAddress" type="string" use="optional" />
593 </complexType>

```

## 2.4.4. Element <AuthorizationDecisionStatement>

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

Resource [Optional]

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

Decision [Optional]

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

<Actions> [Required]

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

<Evidence> [Zero or more]

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

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

```

611     <element name="AuthorizationDecisionStatement"
612     type="saml:AuthorizationDecisionStatementType" />
613     <complexType name="AuthorizationDecisionStatementType">
614         <complexContent>
615             <extension base="saml:SubjectStatementAbstractType">
616                 <sequence>
617                     <element ref="saml:Actions" />
618                     <element ref="saml:Evidence" minOccurs="0"
619                         maxOccurs="unbounded" />
620                 </sequence>
621                 <attribute name="Resource" type="anyURI" use="optional" />
622                 <attribute name="Decision" type="saml:DecisionType"
623                     use="optional" />
624             </extension>
625         </complexContent>
626     </complexType>

```

### 2.4.4.1. Elements <Actions> and <Action>

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

Namespace [Optional]

A URI representing the namespace in which the names of specified actions are to be interpreted. If this element is absent, the namespace <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/rwedc-negation> specified in section 6.2.2 is in effect by default.

<Action> [One or more]

An action sought to be performed on the specified resource.

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

```

<element name="Actions" type="saml:ActionsType" />

```

```

640     <complexType name="ActionTypes">
641       <sequence>
642         <element ref="saml:Action" maxOccurs="unbounded"/>
643       </sequence>
644       <attribute name="Namespace" type="anyURI" use="optional"/>
645     </complexType>
646     <element name="Action" type="string"/>

```

#### 647 2.4.4.2. Element <Evidence>

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

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

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

```

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

```

#### 656 2.4.5. Element <AttributeStatement>

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

660 <Attribute> [One or More]

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

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

```

664     <element name="AttributeStatement" type="saml:AttributeStatementType"/>
665     <complexType name="AttributeStatementType">
666       <complexContent>
667         <extension base="saml:SubjectStatementAbstractType">
668           <sequence>
669             <element ref="saml:Attribute" maxOccurs="unbounded"/>
670           </sequence>
671         </extension>
672       </complexContent>
673     </complexType>

```

##### 674 2.4.5.1. Elements <AttributeDesignator> and <Attribute>

675 The <AttributeDesignator> element identifies an attribute name within an attribute  
676 namespace. It has the **AttributeDesignatorType** complex type. It is used in an attribute  
677 assertion query to request that attribute values within a specific namespace be returned (see  
678 3.4.4 for more information). The <AttributeDesignator> element contains the following XML  
679 attributes:

680 AttributeNamespace [Required]

681 The namespace in which the AttributeName elements are interpreted.

682 AttributeName [Required]

683 The name of the attribute.

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

```

686     <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>

```

```

687 <complexType name="AttributeDesignatorType">
688   <attribute name="AttributeName" type="string"/>
689   <attribute name="AttributeNamespace" type="anyURI"/>
690 </complexType>

```

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

```

694 <AttributeValue> [Required]
695     The value of the attribute.

```

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

```

698 <element name="Attribute" type="saml:AttributeType"/>
699 <complexType name="AttributeType">
700   <complexContent>
701     <extension base="saml:AttributeDesignatorType">
702       <sequence>
703         <element ref="saml:AttributeValue"/>
704       </sequence>
705     </extension>
706   </complexContent>
707 </complexType>

```

#### 708 2.4.5.1.1 Element <AttributeValue>

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

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

```

714 <element name="AttributeValue" type="saml:AttributeValueType"/>
715 <complexType name="AttributeValueType">
716   <sequence>
717     <any namespace="##any" processContents="lax"
718       minOccurs="0" maxOccurs="unbounded"/>
719   </sequence>
720 </complexType>

```

## 3. SAML Protocol

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

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

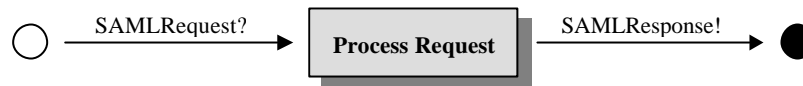


Figure 1: SAML Request-Response Protocol

### 3.1. Schema Header and Namespace Declarations

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

```
<schema
  targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-22.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-22.xsd"
  xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-22.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified">
  <import namespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-22.xsd"
    schemaLocation="draft-sstc-schema-assertion-22.xsd" />
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd" />
  <annotation>
    <documentation>draft-sstc-schema-protocol-22.xsd</documentation>
  </annotation>
...
</schema>
```

### 3.2. Simple Types

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

#### 3.2.1. Simple Type StatusCodeType

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

Success

The request succeeded.

Failure

The request could not be performed by the service.

Error

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

764 Unknown  
765 The request failed for unknown reasons.  
766 The following schema fragment defines the **StatusCodeType** simple type:

```
767 <simpleType name="StatusCodeType">  
768   <restriction base="string">  
769     <enumeration value="Success"/>  
770     <enumeration value="Failure"/>  
771     <enumeration value="Error"/>  
772     <enumeration value="Unknown"/>  
773   </restriction>  
774 </simpleType>
```

## 775 3.3. Requests

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

### 777 3.3.1. Complex Type RequestAbstractType

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

780 RequestID [Required]  
781 An identifier for the request. It is of type **IDType**, and MUST follow the requirements  
782 specified by that type for identifier uniqueness. The values of the RequestID attribute in  
783 a request and the InResponseTo attribute in the corresponding response MUST match.

784 MajorVersion [Required]  
785 The major version of this request. The identifier for the version of SAML defined in this  
786 specification is 1. Processing of this attribute is specified in Section 3.5.2.

787 MinorVersion [Required]  
788 The minor version of this request. The identifier for the version of SAML defined in this  
789 specification is 0. Processing of this attribute is specified in Section 3.5.2.

790 <RespondWith> [Any Number]  
791 Each <RespondWith> element specifies a type of response that is acceptable to the  
792 requestor.

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

```
794 <complexType name="RequestAbstractType" abstract="true">  
795   <sequence>  
796     <element ref="samlp:RespondWith"  
797       minOccurs="0" maxOccurs="unbounded"/>  
798   </sequence>  
799   <attribute name="RequestID" type="saml:IDType" use="required"/>  
800   <attribute name="MajorVersion" type="integer" use="required"/>  
801   <attribute name="MinorVersion" type="integer" use="required"/>  
802 </complexType>
```

#### 803 3.3.1.1. Element <RespondWith>

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

807 SingleStatement  
808 An assertion carrying exactly one statement element.

809 `MultipleStatement`  
 810     An assertion carrying at least one statement element.

811 `AuthenticationStatement`  
 812     An assertion carrying an Authentication statement.

813 `AuthorizationDecisionStatement`  
 814     An assertion carrying an Authorization Decision statement.

815 `AttributeStatement`  
 816     An assertion carrying an Attribute statement.

817 *Schema URI*  
 818     An assertion containing additional elements from the specified schema.

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

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

### 3.3.2. Element `<Request>`

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

826 `<Query>`  
 827     An extension point that allows extension schemas to define new types of query.

828 `<SubjectQuery>`  
 829     An extension point that allows extension schemas to define new types of query that  
 830     specify a single SAML subject.

831 `<AuthenticationQuery>`  
 832     Makes a query for authentication information.

833 `<AttributeQuery>`  
 834     Makes a query for attribute information.

835 `<AuthorizationDecisionQuery>`  
 836     Makes a query for an authorization decision.

837 `<AssertionID>` [One or more]  
 838     Requests an assertion by reference to its assertion identifier.

839 `<AssertionArtifact>` [One or more]  
 840     Requests an assertion by supplying an assertion artifact that represents it.

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

```
<element name="Request" type="samlp:RequestType"/>
<complexType name="RequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <choice>
        <element ref="samlp:Query"/>
        <element ref="samlp:SubjectQuery"/>
        <element ref="samlp:AuthenticationQuery"/>
        <element ref="samlp:AttributeQuery"/>
        <element ref="samlp:AuthorizationDecisionQuery"/>
        <element ref="saml:AssertionID" maxOccurs="unbounded"/>
        <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

```

857     </complexContent>
858   </complexType>
859   <element name="AssertionArtifact" type="string"/>

```

## 3.4. Queries

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

### 3.4.1. Element <Query>

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

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

```

869   <element name="Query" type="samlp:QueryAbstractType"/>
870   <complexType name="QueryAbstractType" abstract="true"/>

```

### 3.4.2. Element <SubjectQuery>

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

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

```

878   <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
879   <complexType name="SubjectQueryAbstractType" abstract="true">
880     <complexContent>
881       <extension base="samlp:QueryAbstractType">
882         <sequence>
883           <element ref="saml:Subject"/>
884         </sequence>
885       </extension>
886     </complexContent>
887   </complexType>

```

### 3.4.3. Element <AuthenticationQuery>

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

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

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



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

```
<element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
<complexType name="AuthenticationQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <sequence>
        <element ref="saml:ConfirmationMethod" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

#### 3.4.4. Element <AttributeQuery>

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

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

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

```
<element name="AttributeQuery" type="samlp:AttributeQueryType"/>
<complexType name="AttributeQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <sequence>
        <element ref="saml:AttributeDesignator"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="CompletenessSpecifier"
        type="samlp:CompletenessSpecifierType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

#### 3.4.5. Element <AuthorizationDecisionQuery>

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

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

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

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

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



```

952     <element name="AuthorizationDecisionQuery"
953     type="samlp:AuthorizationDecisionQueryType"/>
954     <complexType name="AuthorizationDecisionQueryType">
955         <complexContent>
956             <extension base="samlp:SubjectQueryAbstractType">
957                 <sequence>
958                     <element ref="saml:Actions"/>
959                     <element ref="saml:Evidence"
960                         minOccurs="0" maxOccurs="unbounded" />
961                 </sequence>
962                 <attribute name="Resource" type="anyURI" />
963             </extension>
964         </complexContent>
965     </complexType>

```

## 3.5. Responses

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

### 3.5.1. Complex Type ResponseAbstractType

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

**ResponseID** [Required]

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

**InResponseTo** [Required]

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

**MajorVersion** [Required]

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

**MinorVersion** [Required]

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

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

```

984     <complexType name="ResponseAbstractType" abstract="true">
985         <attribute name="ResponseID" type="saml:IDType" use="required"/>
986         <attribute name="InResponseTo" type="saml:IDType" use="required"/>
987         <attribute name="MajorVersion" type="integer" use="required"/>
988         <attribute name="MinorVersion" type="integer" use="required"/>
989     </complexType>

```

### 3.5.2. Element <Response>

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

**StatusCode** [Required] (see Section 3.2.1)

A code representing the status of the corresponding request.

<Assertion> (see Section 2.3.3)

Specifies an assertion by value.

999 <SingleAssertion>  
1000 Specifies an assertion containing a single statement by value.  
1001 <MultipleAssertion>  
1002 Specifies an assertion containing multiple statements by value.  
1003 The following schema fragment defines the <Response> element and its **ResponseType**  
1004 complex type:

```
1005 <element name="Response" type="samlp:ResponseType" />  
1006 <complexType name="ResponseType">  
1007   <complexContent>  
1008     <extension base="samlp:ResponseAbstractType">  
1009       <sequence>  
1010         <element ref="samlp:StatusReason"  
1011           minOccurs="0" maxOccurs="unbounded" />  
1012         <element ref="saml:Assertion"  
1013           minOccurs="0" maxOccurs="unbounded" />  
1014       </sequence>  
1015       <attribute name="StatusCode"  
1016         type="samlp:StatusCodeType" use="required" />  
1017     </extension>  
1018   </complexContent>  
1019 </complexType>
```

#### 1020 3.5.2.1. Element <StatusReason>

1021 The <StatusReason> element provides additional information that indicates the reason for the  
1022 return of an Error or Failure status code. The following values are defined. Implementations  
1023 MAY define additional codes:

1024 RequestVersionTooHigh  
1025 The protocol version specified in the request is a major upgrade from the highest protocol  
1026 version supported by the responder.

1027 RequestVersionTooLow  
1028 The responder cannot respond to the particular request using the SAML version specified  
1029 in the request because it is too low.

1030 RequestVersionDeprecated  
1031 The responder does not respond to any requests with the protocol version specified in  
1032 the request.

1033 TooManyResponses  
1034 The response would contain more elements than the responder will return.

1035 The following schema fragment defines the <StatusReason> element:

```
1036 <element name="StatusReason" type="string" />
```

## 4. SAML Versioning

SAML version information appears in the following elements:

?? <Assertion>

?? <Request>

?? <Response>

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

The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

$Major_B > Major_A \text{ ? } ( ( Major_B = Major_A ) \text{ ? } Minor_B = Minor_A )$

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

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

?? **Documentation change:**  $( Major_B = Major_A ) \text{ ? } ( Minor_B = Minor_A )$

If the major and minor version numbers are unchanged, the new version *B* only introduces changes to the documentation that raise no compatibility issues with an implementation of version *A*.

?? **Minor upgrade:**  $( Major_B = Major_A ) \text{ ? } ( Minor_B > Minor_A )$

If the major version number of versions *A* and *B* are the same and the minor version number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the SAML schema and semantics but any changes that are introduced in *B* SHALL be compatible with version *A*.

?? **Major upgrade:**  $Major_B > Major_A$

If the major version of *B* number is higher than the major version of *A*, Version *B* MAY introduce changes to the SAML schema and semantics that are incompatible with *A*.

### 4.1. Assertion Version

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

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

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

### 4.2. Request Version

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

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

### 4.3. Response Version

1075

1076 A SAML application MUST NOT issue responses that specify a higher SAML version number  
1077 than the corresponding request.

1078 A SAML application MUST NOT issue a response that has a major version number that is lower  
1079 than the major version number of the corresponding request except to report the error  
1080 `RequestVersionTooHigh`.

1081 Incompatible protocol versions MAY cause the following errors to be reported:

1082 `RequestVersionTooHigh`

1083 The protocol version specified in the request is a major upgrade from the highest protocol  
1084 version supported by the responder.

1085 `RequestVersionTooLow`

1086 The responder cannot respond to the particular request using the SAML version specified  
1087 in the request because it is too low.

1088 `RequestVersionDeprecated`

1089 The responder does not respond to any requests with the protocol version specified in  
1090 the request.

## 5. SAML Extensions

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

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

### 5.1. Assertion Schema Extension

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

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

?? <Assertion>

?? <Condition>

?? <Statement>

?? <SubjectStatement>

?? <AdviceElement>

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

?? <SingleAssertion>

?? <MultipleAssertion>

?? <AuthenticationStatement>

?? <AuthorizationDecisionStatement>

?? <AttributeStatement>

?? <AudienceRestrictionCondition>

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

?? <AttributeValue>

?? <Advice>

### 5.2. Protocol Schema Extension

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

?? <Query>

1128        ?? <SubjectQuery>

1129     In addition, the following elements that are directly usable as part of SAML MAY be extended:

1130        ?? <Request>

1131        ?? <AuthenticationQuery>

1132        ?? <AuthorizationDecisionQuery>

1133        ?? <AttributeQuery>

1134        ?? <Response>

### 1135     5.3. Use of Type Derivation and Substitution Groups

1136     W3C XML Schema [Schema1] provides two principal mechanisms for specifying an element of  
1137     an extended type: type derivation and substitution groups.

1138     For example, a <Statement> element can be assigned the type **NewStatementType** by means  
1139     of the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs  
1140     to be derived from **StatementType**. The following example of a SAML assertion assumes that the  
1141     extension schema (represented by the `new:` prefix) has defined this new type:

```
1142     <saml:Assertion ...>  
1143       <saml:Statement xsi:type="new:NewStatementType">  
1144         ...  
1145       </saml:Statement>  
1146     </saml:Assertion>
```

1147     Alternatively, the extension schema can define a <NewStatement> element that is a member of  
1148     a substitution group that has <Statement> as a head element. For the substituted element to be  
1149     schema-valid, it needs to have a type that matches or is derived from the head element's type.  
1150     The following is an example of an extension schema fragment that defines this new element:

```
1151     <xsd:element "NewStatement" type="new:NewStatementType"  
1152       substitutionGroup="saml:Statement" />
```

1153     The substitution group declaration allows the <NewStatement> element to be used anywhere  
1154     the SAML <Statement> element can be used. The following is an example of a SAML assertion  
1155     that uses the extension element:

```
1156     <saml:Assertion ...>  
1157       <new:NewStatement>  
1158         ...  
1159       </new:NewStatement>  
1160     </saml:Assertion>
```

1161     The choice of extension method has no effect on the semantics of the XML document but does  
1162     have implications for interoperability.

1163     The advantages of type derivation are as follows:

1164        ?? A document can be more fully interpreted by a parser that does not have access to the  
1165        extension schema because a "native" SAML element is available.

1166        ?? At the time of writing, some W3C XML Schema validators do not support substitution  
1167        groups, whereas the `xsi:type` attribute is widely supported.

1168     The advantage of substitution groups is that a document can be explained without the need to  
1169     explain the functioning of the `xsi:type` attribute.

## 6. SAML-Defined Identifiers

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

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

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

### 6.1. Confirmation Method Identifiers

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

#### 6.1.1. SAML Artifact:

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

<SubjectConfirmationData>: *Base64 ( Artifact )*

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

#### 6.1.2. SAML Artifact (SHA-1):

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

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

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

#### 6.1.3. Holder of Key:

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

<ds:KeyInfo>: Any cryptographic key

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

#### 6.1.4. Sender Vouches:

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

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

#### 6.1.5. Password (Pass-Through):

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

<SubjectConfirmationData>: *Base64 ( Password )*

1202 The subject of the assertion is the party that can present the password value specified in  
1203 <SubjectConfirmationData>.

1204 The username of the subject is specified by means of the <NameIdentifier> element.

#### 1205 **6.1.6. Password (One-Way-Function SHA-1):**

1206 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/password-sha1>

1207 <SubjectConfirmationData>: *Base64 ( SHA1 ( Password ))*

1208 The subject of the assertion is the party that can present the password such that the SHA1 digest  
1209 of the specified password matches the value specified in <SubjectConfirmationData>.

1210 The username of the subject is specified by means of the <NameIdentifier> element.

#### 1211 **6.1.7. Kerberos [Kerberos]**

1212 **URI:** <urn:ietf:rfc:1510>

1213 <SubjectConfirmationData>: A Kerberos Ticket

#### 1214 **6.1.8. SSL/TLS Certificate Based Client Authentication:**

1215 **URI:** <urn:ietf:rfc:2246>

1216 <ds:KeyInfo>: Any cryptographic key

#### 1217 **6.1.9. Object Authenticator (SHA-1):**

1218 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/object-sha1>

1219 <SubjectConfirmationData>: *Base64 ( SHA1 ( Object ))*

1220 This authenticator element is the result of computing a digest, using the SHA -1 hash algorithm. It  
1221 is used when the subject can be represented as a binary string, for example when it is an XML  
1222 document or the disk image of executable code. Any preprocessing of the subject prior to  
1223 computation of the digest is out of scope. The name of the subject should be conveyed in an  
1224 accompanying NameIdentifier element.

#### 1225 **6.1.10. PKCS#7**

1226 **URI:** <urn:ietf:rfc:2315>

1227 <SubjectConfirmationData>: *Base64 ( PKCS#7 ( Object ))*

1228 This authenticator element is signed data in PKCS#7 format [PKCS#7]. The posited identity of  
1229 the signer must be conveyed in an accompanying NameIdentifier element. This subject type may  
1230 be included in the subject field of an authentication query, in which case the corresponding  
1231 response indicates whether the posited signer is, indeed, the signer. It may be included in an  
1232 attribute query, in which case, the requested attribute values for the subject authenticated by the  
1233 signed data are returned. It may be included in an authorization query, in which case, the access  
1234 request represented by the signed data shall be identified by the accompanying object element,  
1235 and the corresponding authorization decision assertion indicates whether the signer is authorized  
1236 for the access request represented by the object element.



### 6.1.11. Cryptographic Message Syntax

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

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

This authenticator element is signed data in CMS format [CMS]. See also 6.1.10

### 6.1.12. XML Digital Signature

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

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

<ds:KeyInfo>: A cryptographic signing key

This authenticator element is signed data in XML Signature format. See also 6.1.10

## 6.2. Action Namespace Identifiers

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

### 6.2.1. Read/Write/Execute/Delete/Control:

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

Defined actions:

Read Write Execute Delete Control

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

Read

The subject may read the resource

Write

The subject may modify the resource

Execute

The subject may execute the resource

Delete

The subject may delete the resource

Control

The subject may specify the access control policy for the resource

### 6.2.2. Read/Write/Execute/Delete/Control with Negation:

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

Defined actions:

Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

The actions specified in section 6.2.1 are interpreted in the same manner described there. Actions prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action ~Read is affirmatively denied read permission.

1272 An application MUST NOT authorize both an action and its negated form.

### 1273 **6.2.3. Get/Head/Put/Post:**

1274 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/ghpp>

1275 Defined actions:

1276 GET HEAD PUT POST

1277 These actions bind to the corresponding HTTP operations. For example a subject authorized to  
1278 perform the GET action on a resource is authorized to retrieve it.

1279 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT  
1280 and POST actions to the write permission. The correspondence is not exact however since a  
1281 HTTP GET operation may cause data to be modified and a POST operation may cause  
1282 modification to a resource other than the one specified in the request. For this reason a separate  
1283 Action URI specifier is provided.

### 1284 **6.2.4. UNIX File Permissions:**

1285 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-22/unix>

1286 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
1287 notation.

1288 The action string is a four digit numeric code:

1289 *extended user group world*

1290 Where the *extended* access permission has the value

1291 +2 if sgid is set

1292 +4 if suid is set

1293 The *user group* and *world* access permissions have the value

1294 +1 if execute permission is granted

1295 +2 if write permission is granted

1296 +4 if read permission is granted

1297 For example 0754 denotes the UNIX file access permission: user read, write and execute, group  
1298 read and execute and world read.

## 7. SAML Schema Listings

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

### 7.1. Assertion Schema

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

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

```

```

1356 <complexType name="ConditionsType">
1357   <choice minOccurs="0" maxOccurs="unbounded">
1358     <element ref="saml:Condition"/>
1359     <element ref="saml:AudienceRestrictionCondition"/>
1360   </choice>
1361   <attribute name="NotBefore" type="dateTime" use="optional"/>
1362   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
1363 </complexType>
1364 <element name="Condition" type="saml:ConditionAbstractType"/>
1365 <complexType name="ConditionAbstractType" abstract="true"/>
1366 <element name="AudienceRestrictionCondition"
1367   type="saml:AudienceRestrictionConditionType"/>
1368 <complexType name="AudienceRestrictionConditionType">
1369   <complexContent>
1370     <extension base="saml:ConditionAbstractType">
1371       <sequence>
1372         <element ref="saml:Audience" maxOccurs="unbounded"/>
1373       </sequence>
1374     </extension>
1375   </complexContent>
1376 </complexType>
1377 <element name="Audience" type="anyURI"/>
1378 <element name="TargetRestrictionCondition"
1379   type="saml:TargetRestrictionConditionType"/>
1380 <complexType name="TargetRestrictionConditionType">
1381   <complexContent>
1382     <extension base="saml:ConditionAbstractType">
1383       <sequence>
1384         <element ref="saml:Target"
1385           minOccurs="1" maxOccurs="unbounded"/>
1386       </sequence>
1387     </extension>
1388   </complexContent>
1389 </complexType>
1390 <element name="Target" type="anyURI"/>
1391 <element name="Advice" type="saml:AdviceType"/>
1392 <complexType name="AdviceType">
1393   <sequence>
1394     <choice minOccurs="0" maxOccurs="unbounded">
1395       <element ref="saml:AssertionSpecifier"/>
1396       <element ref="saml:AdviceElement"/>
1397       <any namespace="##other" processContents="lax"/>
1398     </choice>
1399   </sequence>
1400 </complexType>
1401 <element name="AdviceElement" type="saml:AdviceAbstractType"/>
1402 <complexType name="AdviceAbstractType"/>
1403 <element name="Statement" type="saml:StatementAbstractType"/>
1404 <complexType name="StatementAbstractType" abstract="true"/>
1405 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1406 <complexType name="SubjectStatementAbstractType" abstract="true">
1407   <complexContent>
1408     <extension base="saml:StatementAbstractType">
1409       <sequence>
1410         <element ref="saml:Subject"/>
1411       </sequence>
1412     </extension>
1413   </complexContent>
1414 </complexType>
1415 <element name="Subject" type="saml:SubjectType"/>
1416 <complexType name="SubjectType">
1417   <choice maxOccurs="unbounded">
1418     <sequence>

```

```

1419         <element ref="saml:NameIdentifier"/>
1420         <element ref="saml:SubjectConfirmation" minOccurs="0"/>
1421     </sequence>
1422     <element ref="saml:SubjectConfirmation"/>
1423 </choice>
1424 </complexType>
1425 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1426 <complexType name="NameIdentifierType">
1427     <attribute name="SecurityDomain" type="string"/>
1428     <attribute name="Name" type="string"/>
1429 </complexType>
1430 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1431 <complexType name="SubjectConfirmationType">
1432     <sequence>
1433         <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1434         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
1435         <element ref="ds:KeyInfo" minOccurs="0"/>
1436     </sequence>
1437 </complexType>
1438 <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
1439 <element name="ConfirmationMethod" type="anyURI"/>
1440 <element name="AuthenticationStatement"
1441     type="saml:AuthenticationStatementType"/>
1442 <complexType name="AuthenticationStatementType">
1443     <complexContent>
1444         <extension base="saml:SubjectStatementAbstractType">
1445             <sequence>
1446                 <element ref="saml:AuthenticationLocality" minOccurs="0"/>
1447             </sequence>
1448             <attribute name="AuthenticationMethod" type="anyURI"/>
1449             <attribute name="AuthenticationInstant" type="dateTime"/>
1450         </extension>
1451     </complexContent>
1452 </complexType>
1453 <element name="AuthenticationLocality"
1454     type="saml:AuthenticationLocalityType"/>
1455 <complexType name="AuthenticationLocalityType">
1456     <attribute name="IPAddress" type="string" use="optional"/>
1457     <attribute name="DNSAddress" type="string" use="optional"/>
1458 </complexType>
1459 <element name="AuthorizationDecisionStatement"
1460     type="saml:AuthorizationDecisionStatementType"/>
1461 <complexType name="AuthorizationDecisionStatementType">
1462     <complexContent>
1463         <extension base="saml:SubjectStatementAbstractType">
1464             <sequence>
1465                 <element ref="saml:Actions"/>
1466                 <element ref="saml:Evidence"
1467                     minOccurs="0" maxOccurs="unbounded"/>
1468             </sequence>
1469             <attribute name="Resource" type="anyURI" use="optional"/>
1470             <attribute name="Decision"
1471                 type="saml:DecisionType" use="optional"/>
1472         </extension>
1473     </complexContent>
1474 </complexType>
1475 <element name="Actions" type="saml:ActionsType"/>
1476 <complexType name="ActionsType">
1477     <sequence>
1478         <element ref="saml:Action" maxOccurs="unbounded"/>
1479     </sequence>
1480     <attribute name="Namespace" type="anyURI" use="optional"/>
1481 </complexType>

```

```

1482 <element name="Action" type="string"/>
1483 <element name="Evidence" type="saml:AssertionSpecifierType"/>
1484 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1485 <complexType name="AttributeStatementType">
1486 <complexContent>
1487 <extension base="saml:SubjectStatementAbstractType">
1488 <sequence>
1489 <element ref="saml:Attribute" maxOccurs="unbounded"/>
1490 </sequence>
1491 </extension>
1492 </complexContent>
1493 </complexType>
1494 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
1495 <complexType name="AttributeDesignatorType">
1496 <attribute name="AttributeName" type="string"/>
1497 <attribute name="AttributeNamespace" type="anyURI"/>
1498 </complexType>
1499 <element name="Attribute" type="saml:AttributeType"/>
1500 <complexType name="AttributeType">
1501 <complexContent>
1502 <extension base="saml:AttributeDesignatorType">
1503 <sequence>
1504 <element ref="saml:AttributeValue"/>
1505 </sequence>
1506 </extension>
1507 </complexContent>
1508 </complexType>
1509 <element name="AttributeValue" type="saml:AttributeValueType"/>
1510 <complexType name="AttributeValueType">
1511 <sequence>
1512 <any namespace="##any" processContents="lax"
1513 minOccurs="0" maxOccurs="unbounded"/>
1514 </sequence>
1515 </complexType>
1516 </schema>
1517

```

## 7.2. Protocol Schema

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

```

1520 <?xml version="1.0" encoding="UTF-8"?>
1521 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1522 (VeriSign Inc.) -->
1523 <schema
1524 targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
1525 sstc-schema-protocol-22.xsd"
1526 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1527 xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1528 schema-assertion-22.xsd"
1529 xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1530 schema-protocol-22.xsd"
1531 xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
1532 <import
1533 namespace="http://www.oasis-open.org/committees/security/docs/draft-
1534 sstc-schema-assertion-22.xsd"
1535 schemaLocation="draft-sstc-schema-assertion-22.xsd"/>
1536 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1537 schemaLocation="xmldsig-core-schema.xsd"/>
1538 <annotation>
1539 <documentation>draft-sstc-schema-protocol-22.xsd</documentation>
1540 </annotation>
1541 <simpleType name="StatusCodeType">

```

```

1542     <restriction base="string">
1543         <enumeration value="Success"/>
1544         <enumeration value="Failure"/>
1545         <enumeration value="Error"/>
1546         <enumeration value="Unknown"/>
1547     </restriction>
1548 </simpleType>
1549 <complexType name="RequestAbstractType" abstract="true">
1550     <sequence>
1551         <element ref="samlp:RespondWith"
1552             minOccurs="0" maxOccurs="unbounded"/>
1553     </sequence>
1554     <attribute name="RequestID" type="saml:IDType" use="required"/>
1555     <attribute name="MajorVersion" type="integer" use="required"/>
1556     <attribute name="MinorVersion" type="integer" use="required"/>
1557 </complexType>
1558 <element name="RespondWith" type="anyURI"/>
1559 <element name="Request" type="samlp:RequestType"/>
1560 <complexType name="RequestType">
1561     <complexContent>
1562         <extension base="samlp:RequestAbstractType">
1563             <choice>
1564                 <element ref="samlp:Query"/>
1565                 <element ref="samlp:SubjectQuery"/>
1566                 <element ref="samlp:AuthenticationQuery"/>
1567                 <element ref="samlp:AttributeQuery"/>
1568                 <element ref="samlp:AuthorizationDecisionQuery"/>
1569                 <element ref="saml:AssertionID" maxOccurs="unbounded"/>
1570                 <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
1571             </choice>
1572         </extension>
1573     </complexContent>
1574 </complexType>
1575 <element name="AssertionArtifact" type="string"/>
1576 <element name="Query" type="samlp:QueryAbstractType"/>
1577 <complexType name="QueryAbstractType" abstract="true">
1578     <sequence>
1579         <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
1580     </sequence>
1581 </complexType>
1582 <complexType name="SubjectQueryAbstractType" abstract="true">
1583     <complexContent>
1584         <extension base="samlp:QueryAbstractType">
1585             <sequence>
1586                 <element ref="saml:Subject"/>
1587             </sequence>
1588         </extension>
1589     </complexContent>
1590 </complexType>
1591 <complexType name="AuthenticationQuery" type="samlp:AuthenticationQueryType">
1592     <complexContent>
1593         <extension base="samlp:SubjectQueryAbstractType">
1594             <sequence>
1595                 <element ref="saml:ConfirmationMethod" minOccurs="0"/>
1596             </sequence>
1597         </extension>
1598     </complexContent>
1599 </complexType>
1600 <complexType name="AttributeQuery" type="samlp:AttributeQueryType">
1601     <complexContent>
1602         <extension base="samlp:SubjectQueryAbstractType">
1603             <sequence>
1604                 <element ref="saml:AttributeDesignator"
1605                     minOccurs="0" maxOccurs="unbounded"/>

```

```

1605         </sequence>
1606     </extension>
1607 </complexContent>
1608 </complexType>
1609 <element name="AuthorizationDecisionQuery"
1610     type="samlp:AuthorizationDecisionQueryType"/>
1611 <complexType name="AuthorizationDecisionQueryType">
1612     <complexContent>
1613         <extension base="samlp:SubjectQueryAbstractType">
1614             <sequence>
1615                 <element ref="saml:Actions"/>
1616                 <element ref="saml:Evidence"
1617                     minOccurs="0" maxOccurs="unbounded"/>
1618             </sequence>
1619             <attribute name="Resource" type="anyURI"/>
1620         </extension>
1621     </complexContent>
1622 </complexType>
1623 <complexType name="ResponseAbstractType" abstract="true">
1624     <attribute name="ResponseID" type="saml:IDType" use="required"/>
1625     <attribute name="InResponseTo" type="saml:IDType" use="required"/>
1626     <attribute name="MajorVersion" type="integer" use="required"/>
1627     <attribute name="MinorVersion" type="integer" use="required"/>
1628 </complexType>
1629 <element name="Response" type="samlp:ResponseType"/>
1630 <element name="Response" type="samlp:ResponseType"/>
1631 <complexType name="ResponseType">
1632     <complexContent>
1633         <extension base="samlp:ResponseAbstractType">
1634             <sequence>
1635                 <element ref="samlp:StatusReason"
1636                     minOccurs="0" maxOccurs="unbounded"/>
1637                 <element ref="saml:Assertion"
1638                     minOccurs="0" maxOccurs="unbounded"/>
1639             </sequence>
1640             <attribute name="StatusCode"
1641                 type="samlp:StatusCodeType" use="required"/>
1642         </extension>
1643     </complexContent>
1644 </complexType>
1645 <element name="StatusReason" type="string"/>
1646 </schema>
1647

```



## 8. References

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

1695	<b>[XMLSig-XSD]</b>	XML Signature Schema available from <a href="http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd">http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd</a>
1696		
1697	<b>[XTAML]</b>	P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> ,
1698		<a href="http://www.xmltrustcenter.org/">http://www.xmltrustcenter.org/</a> , VeriSign Inc. September 2001.

## Appendix A. Notices

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

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

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

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

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

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