



---

# Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

Working Draft 05, 17 February 2004

**Document identifier:**

sstc-saml-core-2.0-draft-05

**Location:**

[http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)

**Editors:**

Scott Cantor, individual ([cantor.2@osu.edu](mailto:cantor.2@osu.edu))  
John Kemp, Nokia ([john.kemp@nokia.com](mailto:john.kemp@nokia.com))  
Eve Maler, Sun Microsystems ([eve.maler@sun.com](mailto:eve.maler@sun.com))

**Contributors:**

Stephen Farrell, Baltimore Technologies  
Irving Reid, Baltimore Technologies  
Hal Lockhart, BEA Systems  
David Orchard, BEA Systems  
Krishna Sankar, Cisco Systems  
Carlisle Adams, Entrust  
Tim Moses, Entrust  
Nigel Edwards, Hewlett-Packard  
Joe Pato, Hewlett-Packard  
Bob Blakley, IBM  
Marlena Erdos, IBM  
RL "Bob" Morgan, individual  
Marc Chanliau, Netegrity  
Chris McLaren, Netegrity  
Prateek Mishra, Netegrity (co-chair)  
Charles Knouse, Oblix  
Simon Godik, Overxeer  
Rob Philpott, RSA Security (co-chair)  
Darren Platt, formerly of RSA Security  
Jahan Moreh, Sigaba  
Jeff Hodges, Sun Microsystems  
Phillip Hallam-Baker, VeriSign (former editor)

37 **Abstract:**

38 This specification defines the syntax and semantics for XML-encoded assertions about  
39 authentication, attributes and authorization, and for the protocols that conveys this information.

40 **Status:**

41 This is a working draft produced by the Security Services Technical Committee. Publication of  
42 this draft does not imply TC endorsement. This is an active working draft that may be updated,  
43 replaced, or obsoleted at any time. **See the Revision History for details of changes made in**  
44 **this revision.**

45 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)  
46 [services@lists.oasis-open.org](mailto:services@lists.oasis-open.org) list. Others should submit them to the [security-services-](mailto:security-services-comment@lists.oasis-open.org)  
47 [comment@lists.oasis-open.org](mailto:comment@lists.oasis-open.org) list (to post, you must subscribe; to subscribe, send a message  
48 to [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with "subscribe" in the body) or use  
49 other OASIS-supported means of submitting comments. The committee will publish vetted errata  
50 on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

51 For information on whether any patents have been disclosed that may be essential to  
52 implementing this specification, and any offers of patent licensing terms, please refer to the  
53 Intellectual Property Rights web page for the Security Services TC ([http://www.oasis-](http://www.oasis-open.org/committees/security/ipr.php)  
54 [open.org/committees/security/ipr.php](http://www.oasis-open.org/committees/security/ipr.php)).

# 55 Table of Contents

56	1 Introduction.....	6
57	1.1 Notation.....	6
58	1.2 Schema Organization and Namespaces.....	7
59	1.2.1 String and URI Values.....	7
60	1.2.2 Time Values.....	7
61	1.2.3 ID and ID Reference Values.....	7
62	1.2.4 Comparing SAML Values.....	8
63	2 SAML Assertions.....	9
64	2.1 Schema Header and Namespace Declarations.....	9
65	2.2 Simple Types.....	9
66	2.2.1 Simple Type DecisionType.....	10
67	2.3 Name Identifiers.....	10
68	2.3.1 Element <BaseNameIdentifier>.....	10
69	2.3.2 Element <NameIdentifier>.....	11
70	2.3.3 Element <EncryptedNameIdentifier>.....	12
71	2.3.4 Element <Issuer>.....	12
72	2.4 Assertions.....	12
73	2.4.1 Element <AssertionIDReference>.....	13
74	2.4.2 Element <Assertion>.....	13
75	2.4.2.0.1 Attributes NotBefore and NotOnOrAfter .....	15
76	2.4.2.0.2 Element <Condition> .....	16
77	2.4.2.0.3 Elements <AudienceRestrictionCondition> and <Audience>.....	16
78	2.4.2.0.4 Element <DoNotCacheCondition>.....	17
79	2.4.2.1 Element <Advice>.....	17
80	2.5 Statements.....	17
81	2.5.1 Element <Statement> .....	18
82	2.5.2 Element <SubjectStatement>.....	18
83	2.5.2.1 Element <Subject> .....	18
84	2.5.2.3 Elements <SubjectConfirmation>, <ConfirmationMethod>, and	
85	<SubjectConfirmationData>.....	19
86	2.5.3 Element <AuthenticationStatement>.....	19
87	2.5.3.1 Element <SubjectLocality>.....	20
88	2.5.4 Element <AttributeStatement>.....	21
89	2.5.4.1 Elements <AttributeDesignator> and <Attribute>.....	21
90	2.5.4.1.1 Element <AttributeValue>.....	22
91	2.5.5 Element <AuthorizationDecisionStatement>.....	22
92	2.5.5.1 Element <Action>.....	23
93	2.5.5.2 Element <Evidence>.....	24
94	3 SAML Protocols.....	25
95	3.1 Schema Header and Namespace Declarations.....	25
96	3.2 Requests and Responses.....	25
97	3.2.1 Complex Type RequestAbstractType.....	26
98	3.2.1.1 Element <RelayState>.....	27
99	3.2.2 Complex Type StatusResponseType.....	27
100	3.2.2.1 Element <Status>.....	28
101	3.2.2.2 Element <StatusCode>.....	29
102	3.2.2.3 Element <StatusMessage>.....	30
103	3.2.2.4 Element <StatusDetail>.....	30

104	3.3 Assertion Query and Request Protocol.....	30
105	3.3.1 Element <AssertionIDRequest>.....	30
106	3.3.2 Element <ArtifactRequest> .....	31
107	3.3.3 Queries.....	31
108	3.3.3.1 Element <SubjectQuery>.....	31
109	3.3.3.2 Element <AuthenticationQuery>.....	32
110	3.3.3.3 Element <AttributeQuery>.....	33
111	3.3.3.4 Element <AuthorizationDecisionQuery>.....	33
112	3.3.4 Element <Response>.....	34
113	3.3.4.1 Responses to Queries.....	35
114	3.4 Federated Name Registration Protocol.....	35
115	3.4.1 Element <RegisterNameIdentifierRequest>.....	35
116	3.4.2 Element <RegisterNameIdentifierResponse>.....	36
117	3.4.3 Processing Rules.....	36
118	3.5 Federation Termination Protocol.....	37
119	3.5.1 Element <FederationTerminationNotification>.....	37
120	3.5.2 Element <FederationTerminationResponse>.....	38
121	3.5.3 Processing Rules.....	38
122	3.6 Single Logout Protocol.....	39
123	3.6.1 Element <LogoutRequest>.....	39
124	3.6.2 Element <LogoutResponse> (UNFINISHED).....	40
125	3.6.3 Processing Rules (UNFINISHED).....	40
126	4 SAML Versioning.....	41
127	4.1 SAML Specification Set Version.....	41
128	4.1.1 Schema Version.....	41
129	4.1.2 SAML Assertion Version.....	41
130	4.1.3 SAML Protocol Version.....	42
131	4.1.3.1 Request Version.....	42
132	4.1.4 Response Version.....	42
133	4.1.5 Permissible Version Combinations.....	43
134	4.2 SAML Namespace Version.....	43
135	4.2.1 Schema Evolution.....	43
136	5 SAML and XML Signature Syntax and Processing.....	44
137	5.1 Signing Assertions.....	45
138	5.2 Request/Response Signing.....	45
139	5.3 Signature Inheritance.....	45
140	5.4 XML Signature Profile.....	45
141	5.4.1 Signing Formats and Algorithms.....	45
142	5.4.2 References.....	45
143	5.4.3 Canonicalization Method.....	46
144	5.4.4 Transforms.....	46
145	5.4.5 KeyInfo.....	46
146	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	46
147	5.4.7 Interoperability with SAML V1.0.....	46
148	5.4.8 Example.....	46
149	6 SAML Extensions.....	49
150	6.1 Assertion Schema Extension.....	49
151	6.2 Protocol Schema Extension.....	49
152	7 SAML-Defined Identifiers.....	51
153	7.1 Authentication Method Identifiers.....	51

154	7.1.1 Password.....	51
155	7.1.2 Kerberos .....	51
156	7.1.3 Secure Remote Password (SRP).....	51
157	7.1.4 Hardware Token.....	52
158	7.1.5 SSL/TLS Certificate Based Client Authentication:.....	52
159	7.1.6 X.509 Public Key .....	52
160	7.1.7 PGP Public Key .....	52
161	7.1.8 SPKI Public Key .....	52
162	7.1.9 XKMS Public Key .....	52
163	7.1.10 XML Digital Signature .....	52
164	7.1.11 Unspecified .....	53
165	7.2 Action Namespace Identifiers.....	53
166	7.2.1 Read/Write/Execute/Delete/Control.....	53
167	7.2.2 Read/Write/Execute/Delete/Control with Negation.....	53
168	7.2.3 Get/Head/Put/Post.....	53
169	7.2.4 UNIX File Permissions.....	54
170	7.3 NameIdentifier Format Identifiers.....	54
171	7.3.1 Unspecified.....	54
172	7.3.2 Email Address.....	55
173	7.3.3 X.509 Subject Name.....	55
174	7.3.4 Windows Domain Qualified Name.....	55
175	7.3.5 Provider Identifier.....	55
176	7.3.6 Federated Identifier.....	55
177	7.3.7 Transient Identifier.....	56
178	8 References.....	57
179	8.1 Normative References.....	57
180	8.2 Non-Normative References.....	57
181		

---

# 1 Introduction

182

183 This specification defines the syntax and semantics for Security Assertion Markup Language (SAML)  
184 assertions and the protocols for requesting and returning them. SAML assertions, requests, and  
185 responses are encoded in XML and use XML namespaces [XMLNS]. They are typically embedded in  
186 other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The  
187 SAML specification for bindings and profiles [SAMLBind] provides frameworks for this embedding and  
188 transport. Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAML-  
189 XSD] are available.

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

## 1.1 Notation

191

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

194 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD  
195 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as  
196 described in IETF RFC 2119 [RFC 2119]:

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

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

202         Listings of SAML schemas appear like this.

203         Example code listings appear like this.

205 In cases of disagreement between the SAML schema documents [SAML-XSD] [SAML-  
206 XSD] and this specification, the schema documents take precedence.

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

- 210 • The prefix `saml:` stands for the SAML assertion namespace,  
211     `urn:oasis:names:tc:SAML:2.0:assertion`.
- 212 • The prefix `samlp:` stands for the SAML request-response protocol namespace,  
213     `urn:oasis:names:tc:SAML:2.0:protocol`.
- 214 • The prefix `ds:` stands for the W3C XML Signature namespace,  
215     <http://www.w3.org/2000/09/xmldsig#> [XMLSig-XSD].
- 216 • The prefix `xenc:` stands for the W3C XML Encryption namespace,  
217     <http://www.w3.org/2001/04/xmlenc#>.
- 218 • The prefix `xsd:` stands for the W3C XML Schema namespace,  
219     <http://www.w3.org/2001/XMLSchema> [Schema1], in example listings. In schema listings, this is  
220     the default namespace and no prefix is shown.

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

## 223 1.2 Schema Organization and Namespaces

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

```
226 urn:oasis:names:tc:SAML:2.0:assertion
```

227 The SAML request-response protocol structures are defined in a schema [SAML-XP] associated with  
228 the following XML namespace:

```
229 urn:oasis:names:tc:SAML:2.0:protocol
```

230 The assertion schema is imported into the protocol schema. Also imported into both schemas is the  
231 schema for XML Signature , which is associated with the following XML namespace:

```
232 http://www.w3.org/2000/09/xmldsig#
```

233 See Section SAML Namespace Version for information on SAML namespace versioning.

### 234 1.2.1 String and URI Values

235 All SAML string and URI reference values have the types **xsd:string** and **xsd:anyURI** respectively,  
236 which are built in to the W3C XML Schema Datatypes specification . All strings in SAML messages  
237 MUST consist of at least one non-whitespace character (whitespace is defined in the XML  
238 Recommendation §2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise  
239 indicated in this specification, all URI reference values MUST consist of at least one non-whitespace  
240 character, and are REQUIRED to be absolute [RFC 2396].

### 241 1.2.2 Time Values

242 All SAML time values have the type **xsd:dateTime**, which is built in to the W3C XML Schema Datatypes  
243 specification , and MUST be expressed in UTC form.

244 SAML system entities SHOULD NOT rely on other applications supporting time resolution finer than  
245 milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

### 246 1.2.3 ID and ID Reference Values

247 The **xsd:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses.  
248 Values declared to be of type **xsd:ID** in this specification MUST satisfy the following properties in  
249 addition to those imposed by the definition of the **xsd:ID** type itself:

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

254 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the  
255 implementation. In the case that a pseudorandom technique is employed, the probability of two randomly  
256 chosen identifiers being identical MUST be less than or equal to  $2^{-128}$  and SHOULD be less than or equal  
257 to  $2^{-160}$ . This requirement MAY be met by encoding a randomly chosen value between 128 and 160 bits in  
258 length. The encoding must conform to the rules defining the **xsd:ID** datatype.

259 The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that  
260 **xsd>IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML  
261 reference identifier might actually be defined in a document separate from that in which the identifier

262 reference is used, which violates the **xsd:IDREF** requirement that its value match the value of an ID  
263 attribute on some element in the same XML document.

## 264 **1.2.4 Comparing SAML Values**

265 Unless otherwise noted, all elements in SAML documents that have the XML Schema **xsd:string** type, or  
266 a type derived from that, **MUST** be compared using an exact binary comparison. In particular, SAML  
267 implementations and deployments **MUST NOT** depend on case-insensitive string comparisons,  
268 normalization or trimming of white space, or conversion of locale-specific formats such as numbers or  
269 currency. This requirement is intended to conform to the W3C Requirements for String Identity,  
270 Matching, and String Indexing [W3C-CHAR].

271 If an implementation is comparing values that are represented using different character encodings, the  
272 implementation **MUST** use a comparison method that returns the same result as converting both values  
273 to the Unicode character encoding, Normalization Form C [UNICODE-C], and then performing an exact  
274 binary comparison. This requirement is intended to conform to the W3C Character Model for the World  
275 Wide Web [W3C-CharMod], and in particular the rules for Unicode-normalized Text.

276 Applications that compare data received in SAML documents to data from external sources **MUST** take  
277 into account the normalization rules specified for XML. Text contained within elements is normalized so  
278 that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the  
279 XML Recommendation §2.11. Attribute values defined as strings (or types derived from strings) are  
280 normalized as described in §3.3.3. All white space characters are replaced with blanks (ASCII code  
281 32<sub>Decimal</sub>).

282 The SAML specification does not define collation or sorting order for attribute or element values. SAML  
283 implementations **MUST NOT** depend on specific sorting orders for values, because these can differ  
284 depending on the locale settings of the hosts involved.



---

## 2 SAML Assertions

285

286 An assertion is a package of information that supplies one or more statements made by a SAML  
287 authority. This SAML specification defines three different kinds of assertion statement that can be  
288 created by a SAML authority. As mentioned above and described in Section SAML Extensions,  
289 extensions are permitted by the SAML assertion schema, allowing user-defined extensions to assertions  
290 and statements, as well as allowing the definition of new kinds of assertion and statement. The three  
291 kinds of statement defined in this specification are:

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

296 The outer structure of an assertion is generic, providing information that is common to all of the  
297 statements within it. Within an assertion, a series of inner elements describe the authentication, attribute,  
298 authorization decision, or user-defined statements containing the specifics.

### 2.1 Schema Header and Namespace Declarations

299

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

```
302 <schema
303     targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
304     xmlns="http://www.w3.org/2001/XMLSchema"
305     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
306     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
307     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
308     elementFormDefault="unqualified"
309     attributeFormDefault="unqualified"
310     blockDefault="substitution"
311     version="2.0">
312     <import namespace="http://www.w3.org/2000/09/xmldsig#"
313           schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
314 schema.xsd"/>
315     <import namespace="http://www.w3.org/2001/04/xmlenc#"
316           schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
317 schema.xsd"/>
318     <annotation>
319         <documentation>
320             Document identifier: sstc-saml-schema-assertion-2.0
321             Location: http://www.oasis-
322 open.org/committees/documents.php?wg_abbrev=security
323             Revision history:
324             V2.0:
325                 Updated the schema and namespace to V2.0.
326                 Removed <AuthorityBinding> and corresponding type.
327                 Added new NameIdentifier and Issuer types and elements.
328         </documentation>
329     </annotation>
330     ...
331 </schema>
```

### 2.2 Simple Types

332

333 The following section defines the SAML assertion-related simple types.

## 334 2.2.1 Simple Type DecisionType

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

337 Permit

338 The specified action is permitted.

339 Deny

340 The specified action is denied.

341 Indeterminate

342 The SAML authority cannot determine whether the specified action is permitted or denied.

343 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability  
344 to provide an affirmative statement that it is not able to issue a decision. Additional information as to the  
345 reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>` elements.

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

```
347 <simpleType name="DecisionType">  
348   <restriction base="string">  
349     <enumeration value="Permit"/>  
350     <enumeration value="Deny"/>  
351     <enumeration value="Indeterminate"/>  
352   </restriction>  
353 </simpleType>
```

## 354 2.3 Name Identifiers

355 The following sections define the SAML constructs that contain descriptive identifiers of subjects and  
356 assertion and message issuers.

### 357 2.3.1 Element `<BaseNameIdentifier>`

358 The `<BaseNameIdentifier>` element is an extension point that allows applications to add new kinds  
359 of name identifiers. Its **BaseNameIdentifierAbstractType** complex type is abstract and is thus usable  
360 only as the base of a derived type. It defines the following common attributes for all name identifier  
361 representations:

362 NameQualifier [Optional]

363 The security or administrative domain that qualifies the name identifier of the subject. This  
364 attribute provides a means to federate names from disparate user stores without collision.

365 SPNameQualifier [Optional]

366 Further qualifies a federated name identifier with the name of the service provider or affiliation of  
367 providers which has federated the principal's identity.

368

369

370 The following schema fragment defines the `<BaseNameIdentifier>` element and its  
371 **BaseNameIdentifierType** complex type:

```
372 <element name="BaseNameIdentifier" type="saml:BaseNameIdentifierAbstractType"/>  
373 <complexType name="BaseNameIdentifierAbstractType" abstract="true">
```

```

374     <complexContent>
375         <extension base="anyType">
376             <attribute name="NameQualifier" type="string" use="optional"/>
377             <attribute name="SPNameQualifier" type="string" use="optional"/>
378         </extension>
379     </complexContent>
380 </complexType>

```

### 381 2.3.2 Element <NameIdentifier>

382 The <NameIdentifier> element is of type **NameIdentifierType**, which restricts  
383 **BaseNameIdentifierAbstractType** to simple string content and provides additional attributes as follows:

384 **Format** [Optional]

385 A URI reference representing the classification of string-based identifier information. See Section  
386 NameIdentifier Format Identifiers for some URI references that MAY be used as the value of the  
387 **Format** attribute and their associated descriptions and processing rules. If no **Format** value is  
388 provided, the identifier urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see Section  
389 Unspecified) is in effect.

390 When a **Format** value other than those specified in Section NameIdentifier Format Identifiers is  
391 used, the content of the <NameIdentifier> element is to be interpreted according to the  
392 specification of that format as defined outside of this specification. If not otherwise indicated by  
393 the specification of the format, issues of anonymity, pseudonymity, and the persistence of the  
394 identifier with respect to the asserting and relying parties are implementation-specific.

395 **SPProvidedIdentifier** [Optional]

396 The name identifier established by the service provider or affiliation of providers for the principal,  
397 if different from the primary name identifier given in the content of the <NameIdentifier>  
398 element.

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

```

401 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
402 <complexType name="NameIdentifierType" mixed="false">
403     <simpleContent>
404         <restriction base="saml:BaseNameIdentifierAbstractType">
405             <simpleType>
406                 <restriction base="string"/>
407             </simpleType>
408             <attribute name="Format" type="anyURI" use="optional"/>
409             <attribute name="SPProvidedIdentifier" type="string"
410 use="optional"/>
411         </restriction>
412     </simpleContent>
413 </complexType>

```

### 414 2.3.3 Element <EncryptedNameIdentifier>

415 The <EncryptedNameIdentifier> element extends **BaseNameIdentifierAbstractType** to carry the  
416 content of the element in encrypted fashion, as defined by the XML Encryption Syntax and Processing  
417 specification. The <EncryptedNameIdentifier> element contains the following additional elements  
418 and attributes:

419 <xenc:EncryptedData> [Required]  
420 The encrypted content and associated encryption details, as defined by . The encrypted content  
421 MUST contain an element that has a type that is derived from  
422 **BaseNameIdentifierAbstractType**.

423 <xenc:EncryptedKey> [Zero or more]  
424 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a  
425 Recipient attribute that specifies the entity for whom the key has been encrypted.

426 Encrypted identifiers are intended as a privacy protection when the plain-text value passes through an  
427 intermediary; as such, the ciphertext MUST be unique to any given encryption operation. For more on  
428 such issues, see §6.3.

429 The following schema fragment defines the <EncryptedNameIdentifier> element and its  
430 **EncryptedNameIdentifierType** complex type:

```
431 <element name="EncryptedNameIdentifier"  
432 type="saml:EncryptedNameIdentifierType"/>  
433 <complexType name="EncryptedNameIdentifierType" mixed="false">  
434 <complexContent>  
435 <restriction base="saml:BaseNameIdentifierType">  
436 <sequence>  
437 <element ref="xenc:EncryptedData"/>  
438 <element ref="xenc:EncryptedKey" minOccurs="0"  
439 maxOccurs="unbounded"/>  
440 </sequence>  
441 </restriction>  
442 </complexContent>  
443 </complexType>
```

### 444 2.3.4 Element <Issuer>

445 The <Issuer> element, with complex type **NameIdentifierType**, provides structured information about  
446 the issuer of a SAML assertion or protocol message. The element requires the use of a string to carry the  
447 issuer's name, but permits various attributes of descriptive metadata.

448 The following schema fragment defines the <Issuer> element:

```
449 <element name="Issuer" type="saml:NameIdentifierType"/>
```

## 450 2.4 Assertions

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

### 452 2.4.1 Element <AssertionIDReference>

453 The <AssertionIDReference> element makes a reference to a SAML assertion.

454 The following schema fragment defines the <AssertionIDReference> element:

```
455 <element name="AssertionIDReference" type="NCName"/>
```

### 456 2.4.2 Element <Assertion>

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

459 MajorVersion [Required]  
460 The major version of this assertion. The identifier for the version of SAML defined in this  
461 specification is 2. SAML versioning is discussed in Section SAML Versioning.

462 MinorVersion [Required]  
463 The minor version of this assertion. The identifier for the version of SAML defined in this  
464 specification is 0. SAML versioning is discussed in Section SAML Versioning.

465 AssertionID [Required]  
466 The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in  
467 Section 1.2.3 for identifier uniqueness.

468 IssueInstant [Required]  
469 The time instant of issue in UTC, as described in Section Time Values.

470 <Issuer> [Required]  
471 The SAML authority that is making the claim(s) in the assertion. The issuer identity SHOULD be  
472 unambiguous to the intended relying parties. If the Format attribute is omitted, the identifier  
473 urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see section 7.3.1) is  
474 assumed.

475 This specification defines no relationship between the entity represented by this element and the  
476 signer of the assertion (if any). Any such requirements imposed by a relying party that consumes the  
477 assertion or to specific profiles are application-specific.

478 <ds:Signature> [Optional]  
479 An XML Signature that authenticates the assertion, as described in Section SAML and XML  
480 Signature Syntax and Processing.

481 <Conditions> [Optional]  
482 Conditions that MUST be taken into account in assessing the validity of the assertion.

483 <Advice> [Optional]  
484 Additional information related to the assertion that assists processing in certain situations but which  
485 MAY be ignored by applications that do not support its use.

486 One or more of the following statement elements:  
487 <Statement>  
488 A statement defined in an extension schema.

489 <SubjectStatement>  
490 A subject statement defined in an extension schema.

491 <AuthenticationStatement>  
492 An authentication statement.

493 <AuthorizationDecisionStatement>  
494 An authorization decision statement.

495 <AttributeStatement>  
496 An attribute statement.

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

```

499 <element name="Assertion" type="saml:AssertionType"/>
500 <complexType name="AssertionType">
501   <sequence>
502     <element ref="saml:Issuer"/>
503     <element ref="ds:Signature" minOccurs="0"/>
504     <element ref="saml:Conditions" minOccurs="0"/>
505     <element ref="saml:Advice" minOccurs="0"/>
506     <choice maxOccurs="unbounded">
507       <element ref="saml:Statement"/>
508       <element ref="saml:SubjectStatement"/>
509       <element ref="saml:AuthenticationStatement"/>
510       <element ref="saml:AuthorizationDecisionStatement"/>
511       <element ref="saml:AttributeStatement"/>
512     </choice>
513   </sequence>
514   <attribute name="MajorVersion" type="integer" use="required"/>
515   <attribute name="MinorVersion" type="integer" use="required"/>
516   <attribute name="AssertionID" type="ID" use="required"/>
517   <attribute name="IssueInstant" type="dateTime" use="required"/>
518 </complexType>Element <Conditions>

```

519 The <Conditions> element MAY contain the following elements and attributes:

520 NotBefore [Optional]

521 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC  
522 as described in Section Time Values.

523 NotOnOrAfter [Optional]

524 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC as  
525 described in Section Time Values.

526 <Condition> [Any Number]

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

528 <AudienceRestrictionCondition> [Any Number]

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

530 <DoNotCacheCondition> [Any Number]

531 Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future  
532 use.

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

```

535 <element name="Conditions" type="saml:ConditionsType"/>
536 <complexType name="ConditionsType">
537   <choice minOccurs="0" maxOccurs="unbounded">
538     <element ref="saml:AudienceRestrictionCondition"/>
539     <element ref="saml:DoNotCacheCondition">
540       <element ref="saml:Condition"/>
541     </choice>
542   <attribute name="NotBefore" type="dateTime" use="optional"/>
543   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
544 </complexType>

```

545 If an assertion contains a <Conditions> element, the validity of the assertion is dependent on the sub-  
546 elements and attributes provided. When processing the sub-elements and attributes of a  
547 <Conditions> element, the following rules MUST be used in the order shown to determine the overall  
548 validity of the assertion:

- 549 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is  
550 considered to be **Valid**.
- 551 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the  
552 assertion is **Invalid**.
- 553 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity  
554 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 555 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then  
556 the assertion is considered to be **Valid**.

557 The <Conditions> element MAY be extended to contain additional conditions. If an element contained  
558 within a <Conditions> element is encountered that is not understood, the status of the condition  
559 cannot be evaluated and the validity status of the assertion MUST be deemed to be **Indeterminate** in  
560 accordance with rule 3 above.

561 Note that an assertion that has validity status **Valid** may not be trustworthy for reasons such as not being  
562 issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

#### 563 **2.4.2.0.1 Attributes NotBefore and NotOnOrAfter**

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

565 The `NotBefore` attribute specifies the time instant at which the validity interval begins. The  
566 `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.

567 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the  
568 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the  
569 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the  
570 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),  
571 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither  
572 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is  
573 valid at any time.

574 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is  
575 built in to the W3C XML Schema Datatypes specification . All time instants are specified in Universal  
576 Coordinated Time (UTC) as described in Section Time Values.

577 Implementations MUST NOT generate time instants that specify leap seconds.

#### 578 **2.4.2.0.2 Element <Condition>**

579 The <Condition> element serves as an extension point for new conditions. Its  
580 **ConditionAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

581 The following schema fragment defines the <Condition> element and its **ConditionAbstractType**  
582 complex type:

```
583 <element name="Condition" type="saml:ConditionAbstractType"/>  
584 <complexType name="ConditionAbstractType" abstract="true"/>
```

#### 585 **2.4.2.0.3 Elements <AudienceRestrictionCondition> and <Audience>**

586 The <AudienceRestrictionCondition> element specifies that the assertion is addressed to one or  
587 more specific audiences identified by <Audience> elements. Although a SAML relying party that is  
588 outside the audiences specified is capable of drawing conclusions from an assertion, the SAML authority

589 explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the  
590 following elements:

591 <Audience>

592 A URI reference that identifies an intended audience. The URI reference MAY identify a document  
593 that describes the terms and conditions of audience membership.

594 The audience restriction condition evaluates to **Valid** if and only if the SAML relying party is a member of  
595 one or more of the audiences specified.

596 The SAML authority cannot prevent a party to whom the assertion is disclosed from taking action on the  
597 basis of the information provided. However, the <AudienceRestrictionCondition> element allows  
598 the SAML authority to state explicitly that no warranty is provided to such a party in a machine- and  
599 human-readable form. While there can be no guarantee that a court would uphold such a warranty  
600 exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably  
601 improved.

602 The following schema fragment defines the <AudienceRestrictionCondition> element and its  
603 **AudienceRestrictionConditionType** complex type:

```
604 <element name="AudienceRestrictionCondition"  
605 type="saml:AudienceRestrictionConditionType"/>  
606 <complexType name="AudienceRestrictionConditionType">  
607 <complexContent>  
608 <extension base="saml:ConditionAbstractType">  
609 <sequence>  
610 <element ref="saml:Audience" maxOccurs="unbounded"/>  
611 </sequence>  
612 </extension>  
613 </complexContent>  
614 </complexType>  
615 <element name="Audience" type="anyURI"/>
```

#### 616 2.4.2.0.4 Element <DoNotCacheCondition>

617 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be  
618 retained for future use. A SAML authority SHOULD NOT include more than one  
619 <DoNotCacheCondition> element within a <Conditions> element of an assertion. Note that no  
620 Relying Party implementation is required to perform caching. However, any that do so MUST observe  
621 this condition. If multiple <DoNotCacheCondition> elements appear within a <Conditions> element,  
622 a Relying Party MUST treat the multiple elements as though a single <DoNotCacheCondition>  
623 element was specified. For the purposes of determining the validity of the <Conditions> element, the  
624 <DoNotCacheCondition> (see Section Element <Conditions>) is considered to always be valid.

625

```
626 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType" />  
627 <complexType name="DoNotCacheConditionType">  
628 <complexContent>  
629 <extension base="saml:ConditionAbstractType"/>  
630 </complexContent>  
631 </complexType>
```

#### 632 2.4.2.1 Element <Advice>

633 The <Advice> element contains any additional information that the SAML authority wishes to provide.  
634 This information MAY be ignored by applications without affecting either the semantics or the validity of  
635 the assertion.



636 The <Advice> element contains a mixture of zero or more <Assertion> elements,  
637 <AssertionIDReference> elements, and elements in other namespaces, with lax schema validation  
638 in effect for these other elements.

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

- 640 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating  
641 the claims) or indirectly (by reference to the supporting assertions).
- 642 • State a proof of the assertion claims.
- 643 • Specify the timing and distribution points for updates to the assertion.

644 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
645 <element name="Advice" type="saml:AdviceType"/>  
646 <complexType name="AdviceType">  
647   <choice minOccurs="0" maxOccurs="unbounded">  
648     <element ref="saml:AssertionIDReference"/>  
649     <element ref="saml:Assertion"/>  
650     <any namespace="##other" processContents="lax"/>  
651   </choice>  
652 </complexType>
```

## 653 2.5 Statements

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

### 655 2.5.1 Element <Statement>

656 The <Statement> element is an extension point that allows other assertion-based applications to reuse  
657 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable  
658 only as the base of a derived type.

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

```
661 <element name="Statement" type="saml:StatementAbstractType"/>  
662 <complexType name="StatementAbstractType" abstract="true"/>
```

### 663 2.5.2 Element <SubjectStatement>

664 The <SubjectStatement> element is an extension point that allows other assertion-based  
665 applications to reuse the SAML assertion framework. It contains a <Subject> element that allows a  
666 SAML authority to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends  
667 **StatementAbstractType**, is abstract and is thus usable only as the base of a derived type.

668 The **SubjectStatementAbstractType** contains an optional `SessionIndex` attribute.

669 `SessionIndex` [Optional]

670 Indexes a particular session between the subject, and the authority issuing this statement. The value  
671 of the attribute SHOULD be a small, positive integer, but may be any string of text. This value MUST  
672 NOT be a globally unique value for a principal's session at the authority.

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

```
675 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
```

```

676 <complexType name="SubjectStatementAbstractType" abstract="true">
677   <complexContent>
678     <extension base="saml:StatementAbstractType">
679       <sequence>
680         <element ref="saml:Subject"/>
681       </sequence>
682       <attribute name="SessionIndex" type="string"
683 use="optional"/>
684     </extension>
685   </complexContent>
686 </complexType>

```

### 687 2.5.2.1 Element <Subject>

688 The <Subject> element specifies the principal that is the subject of the statement. It contains either or  
689 both of the following elements:

690 <NameIdentifier>, <EncryptedNameIdentifier>, Or <BaseNameIdentifier>

691 An identification of a subject by its name, security domain(s), and other associated information

692 <SubjectConfirmation>

693 Information that allows the subject to be authenticated.

694 If the <Subject> element contains both a <NameIdentifier> and a <SubjectConfirmation>, the  
695 SAML authority is asserting that if the SAML relying party performs the specified  
696 <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the relying  
697 party is the entity that the SAML authority associates with the <NameIdentifier>. A <Subject>  
698 element SHOULD NOT identify more than one principal.

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

```

700 <element name="Subject" type="saml:SubjectType"/>
701 <complexType name="SubjectType">
702   <choice>
703     <sequence>
704       <choice>
705         <element ref="saml:BaseNameIdentifier"/>
706         <element ref="saml:NameIdentifier"/>
707         <element ref="saml:EncryptedNameIdentifier"/>
708       </choice>
709       <element ref="saml:SubjectConfirmation" minOccurs="0"/>
710     </sequence>
711     <element ref="saml:SubjectConfirmation"/>
712   </choice>
713 </complexType>

```

### 714 2.5.2.2

### 715 2.5.2.3 Elements <SubjectConfirmation>, <ConfirmationMethod>, and 716 <SubjectConfirmationData>

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

719 <ConfirmationMethod> [One or more]  
720 A URI reference that identifies a protocol to be used to authenticate the subject. URI references  
721 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the  
722 SAML bindings and profiles specification [SAMLBind]. Additional methods may be added by defining  
723 new profiles or by private agreement.

724 <SubjectConfirmationData> [Optional]  
725 Additional authentication information to be used by a specific authentication protocol.

726 <ds:KeyInfo> [Optional]  
727 An XML Signature element that provides access to a cryptographic key held by the subject.

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

```
731 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
732 <complexType name="SubjectConfirmationType">  
733   <sequence>  
734     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>  
735     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
736     <element ref="ds:KeyInfo" minOccurs="0"/>  
737   </sequence>  
738 </complexType>  
739 <element name="SubjectConfirmationData" type="anyType"/>  
740 <element name="ConfirmationMethod" type="anyURI"/>
```

### 741 2.5.3 Element <AuthenticationStatement>

742 The <AuthenticationStatement> element describes a statement by the SAML authority asserting  
743 that the statement's subject was authenticated by a particular means at a particular time. It is of type  
744 **AuthenticationStatementType**, which extends **SubjectStatementAbstractType** with the addition of the  
745 following elements and attributes:

746 AuthenticationMethod [Required]  
747 A URI reference that specifies the type of authentication that took place. URI references identifying  
748 common authentication protocols are listed in Section Authentication Method Identifiers.

749 AuthenticationInstant [Required]  
750 Specifies the time at which the authentication took place. The time value is encoded in UTC as  
751 described in Section Time Values.

752 <SubjectLocality> [Optional]  
753 Specifies the DNS domain name and IP address for the system entity from which the subject was  
754 apparently authenticated.

755 **Note:** The <AuthorityBinding> element and its corresponding type were removed  
756 from <AuthenticationStatement> for V2.0 of SAML.

757 <AuthenticationStatement> elements **MUST** contain a `SessionIndex` value, conforming to the  
758 rules specified in section 2.5.2.

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

```
761 <element name="AuthenticationStatement"  
762   type="saml:AuthenticationStatementType"/>
```

```

763 <complexType name="AuthenticationStatementType">
764   <complexContent>
765     <extension base="saml:SubjectStatementAbstractType">
766       <sequence>
767         <element ref="saml:SubjectLocality" minOccurs="0"/>
768       </sequence>
769       <attribute name="AuthenticationMethod" type="anyURI"
770 use="required"/>
771       <attribute name="AuthenticationInstant" type="dateTime"
772 use="required"/>
773     </extension>
774   </complexContent>
775 </complexType>

```

### 776 2.5.3.1 Element <SubjectLocality>

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

779 IPAddress [Optional]

780 The IP address of the system entity that was authenticated.

781 DNSAddress [Optional]

782 The DNS address of the system entity that was authenticated.

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

785 The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**  
786 complex type:

```

787 <element name="SubjectLocality"
788   type="saml:SubjectLocalityType"/>
789 <complexType name="SubjectLocalityType">
790   <attribute name="IPAddress" type="string" use="optional"/>
791   <attribute name="DNSAddress" type="string" use="optional"/>
792 </complexType>

```

### 793 2.5.4 Element <AttributeStatement>

794 The <AttributeStatement> element describes a statement by the SAML authority asserting that the  
795 statement’s subject is associated with the specified attributes. It is of type **AttributeStatementType**,  
796 which extends **SubjectStatementAbstractType** with the addition of the following element:

797 <Attribute> [One or More]

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

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

```

801 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
802 <complexType name="AttributeStatementType">
803   <complexContent>
804     <extension base="saml:SubjectStatementAbstractType">
805       <sequence>
806         <element ref="saml:Attribute"
807 maxOccurs="unbounded"/>
808       </sequence>
809     </extension>
810   </complexContent>

```

811 </complexType>

#### 812 2.5.4.1 Elements <AttributeDesignator> and <Attribute>

813 The <AttributeDesignator> element identifies an attribute name within an attribute namespace. It  
814 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that attribute  
815 values within a specific namespace be returned (see Section Element <AttributeQuery> for more  
816 information). The <AttributeDesignator> element contains the following XML attributes:

817 AttributeNamespace [Required]

818 The namespace in which the AttributeName elements are interpreted.

819 AttributeName [Required]

820 The name of the attribute.

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

```
823 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
824 <complexType name="AttributeDesignatorType">
825   <attribute name="AttributeName" type="string" use="required"/>
826   <attribute name="AttributeNamespace" type="anyURI" use="required"/>
827 </complexType>
```

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

831 <AttributeValue> [Any Number]

832 The value of the attribute. If an attribute contains more than one discrete value, it is  
833 RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute  
834 exists but has no value, then the <AttributeValue> element MUST be omitted.

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

```
836 <element name="Attribute" type="saml:AttributeType"/>
837 <complexType name="AttributeType">
838   <complexContent>
839     <extension base="saml:AttributeDesignatorType">
840       <sequence>
841         <element ref="saml:AttributeValue" minOccurs="0"
842           maxOccurs="unbounded"/>
843       </sequence>
844     </extension>
845   </complexContent>
846 </complexType>
```

##### 847 2.5.4.1.1 Element <AttributeValue>

848 The <AttributeValue> element supplies the value of a specified attribute. It is of the **anyType** type,  
849 which allows any well-formed XML to appear as the content of the element.

850 If the data content of an AttributeValue element is of an XML Schema simple type (such as **xsd:integer**  
851 or **xsd:string**), the data type MAY be declared explicitly by means of an **xsi:type** declaration in the  
852 <AttributeValue> element. If the attribute value contains structured data, the necessary data  
853 elements MAY be defined in an extension schema.

854 The following schema fragment defines the <AttributeValue> element:

```
855 <element name="AttributeValue" type="anyType"/>
```

## 856 2.5.5 Element <AuthorizationDecisionStatement>

857 The <AuthorizationDecisionStatement> element describes a statement by the SAML authority  
858 asserting that a request for access by the statement's subject to the specified resource has resulted in  
859 the specified authorization decision on the basis of some optionally specified evidence.

860 The resource is identified by means of a URI reference. In order for the assertion to be interpreted  
861 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference  
862 in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different  
863 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
864 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

865         In general, the rules for equivalence and definition of a normal form, if any, are scheme  
866         dependent. When a scheme uses elements of the common syntax, it will also use the common  
867         syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
868         with an explicit ":port", where the port is the default for the scheme, is equivalent to one where  
869         the port is elided.

870 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the  
871 URI normalized form wherever possible as follows:

- 872 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 873 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

874 Inconsistent URI reference interpretation can also result from differences between the URI reference  
875 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
876 employed to specify an access control policy language. The following security conditions should be  
877 satisfied by the system which employs SAML assertions:

- 878 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
879 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
880 part of the resource URI reference.
- 881 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
882 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
883 gain access to a denied resource by creating such an equivalence.

884 The <AuthorizationDecisionStatement> element is of type  
885 **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the  
886 addition of the following elements (in order) and attributes:

887 **Resource** [Required]

888         A URI reference identifying the resource to which access authorization is sought. It is permitted for  
889         this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the  
890         start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

891 **Decision** [Required]

892         The decision rendered by the SAML authority with respect to the specified resource. The value is of  
893         the **DecisionType** simple type.

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

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

896 <Evidence> [Optional]

897 A set of assertions that the SAML authority relied on in making the decision.

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

```
900 <element name="AuthorizationDecisionStatement"  
901 type="saml:AuthorizationDecisionStatementType"/>  
902 <complexType name="AuthorizationDecisionStatementType">  
903 <complexContent>  
904 <extension base="saml:SubjectStatementAbstractType">  
905 <sequence>  
906 <element ref="saml:Action" maxOccurs="unbounded"/>  
907 <element ref="saml:Evidence" minOccurs="0"/>  
908 </sequence>  
909 <attribute name="Resource" type="anyURI" use="required"/>  
910 <attribute name="Decision" type="saml:DecisionType"  
911 use="required"/>  
912 </extension>  
913 </complexContent>  
914 </complexType>
```

### 915 2.5.5.1 Element <Action>

916 The <Action> element specifies an action on the specified resource for which permission is sought. It  
917 has the following attribute and string-data content:

918 Namespace [Optional]

919 A URI reference representing the namespace in which the name of the specified action is to be  
920 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwdc-  
921 negation specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

922 *string data* [Required]

923 An action sought to be performed on the specified resource.

924 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```
925 <element name="Action" type="saml:ActionType"/>  
926 <complexType name="ActionType">  
927 <simpleContent>  
928 <extension base="string">  
929 <attribute name="Namespace" type="anyURI"/>  
930 </extension>  
931 </simpleContent>  
932 </complexType>
```

### 933 2.5.5.2 Element <Evidence>

934 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on  
935 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one  
936 or more of the following elements:

937 <AssertionIDReference> [Any number]

938 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

939 <Assertion> [Any number]

940 Specifies an assertion by value.

941 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party  
942 and the SAML authority making the authorization decision. For example, in the case that the SAML  
943 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use  
944 that assertion as evidence in making its authorization decision without endorsing the <Evidence>  
945 element's assertion as valid either to the relying party or any other third party.

946 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
947 <element name="Evidence" type="saml:EvidenceType"/>  
948 <complexType name="EvidenceType">  
949   <choice maxOccurs="unbounded">  
950     <element ref="saml:AssertionIDReference"/>  
951     <element ref="saml:Assertion"/>  
952   </choice>  
953 </complexType>
```



## 3 SAML Protocols

954

955 SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of  
956 protocols. The bindings and profiles specification for SAML [SAMLBind] describes specific means of  
957 transporting queries, assertions, and other messages using existing widely deployed protocols.

958 SAML request and response messages derive from common abstract types, described below. The  
959 requester sends an element derived from **RequestAbstractType** to a SAML responder, and the  
960 responder generates an element derived from **StatusResponseType**, as shown in Figure 1. Each  
961 request element typically corresponds to a specific kind of response element.



962

963

Figure 1: SAML Request-Response Protocol

### 3.1 Schema Header and Namespace Declarations

964

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

```
967 <schema  
968   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
969   xmlns="http://www.w3.org/2001/XMLSchema"  
970   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
971   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
972   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
973   elementFormDefault="unqualified"  
974   attributeFormDefault="unqualified"  
975   blockDefault="substitution"  
976   version="2.0">  
977   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
978     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>  
979   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
980     schemaLocation="http://www.w3.org/TR/xmldsig-core/  
981 schema.xsd "/>  
982   <annotation>  
983     <documentation>  
984       Document identifier: sstc-saml-schema-protocol-2.0  
985       Location: http://www.oasis-  
986 open.org/committees/documents.php?wg_abbrev=security  
987       Revision history:  
988         Updated the schema and namespace to V2.0.  
989         Removed <RespondWith> and its corresponding type.  
990         Added Issuer element, revamped protocol types, added RNI  
991 messages.  
992     </documentation>  
993   </annotation>  
994   ...  
995 </schema>
```

### 3.2 Requests and Responses

996

997 The following sections define the SAML constructs that underlie request and response messages.

### 998 3.2.1 Complex Type RequestAbstractType

999 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
1000 This type defines common attributes and elements that are associated with all SAML requests:

1001 RequestID [Required]

1002 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in  
1003 Section 1.2.3 for identifier uniqueness. The values of the RequestID attribute in a request and the  
1004 InResponseTo attribute in the corresponding response MUST match.

1005 MajorVersion [Required]

1006 The major version of this request. The identifier for the version of SAML defined in this specification  
1007 is 2. SAML versioning is discussed in Section SAML Versioning.

1008 MinorVersion [Required]

1009 The minor version of this request. The identifier for the version of SAML defined in this specification  
1010 is 0. SAML versioning is discussed in Section SAML Versioning.

1011 IssueInstant [Required]

1012 The time instant of issue of the request. The time value is encoded in UTC as described in Section  
1013 Time Values.

1014 Consent [Optional]

1015 Indicates whether or not consent has been obtained from a user in the sending this request.

1016 <Issuer> [Optional]

1017 Identifies the entity that generated the request message.

1018 <ds:Signature> [Optional]

1019 An XML Signature that authenticates the request, as described in Section SAML and XML Signature  
1020 Syntax and Processing.

1021 <RelayState> [Optional]

1022 This contains state information that MUST be relayed back in the associated response.

1023 <Extensions> [Optional]

1024 This contains optional protocol message extension elements that are agreed upon between the  
1025 communicating parties.

1026 **Note:** The <RespondWith> element has been removed from <Request> for V2.0 of  
1027 SAML.

1028 The following schema fragment defines the **RequestAbstractType** complex type:

```
1029 <complexType name="RequestAbstractType" abstract="true">  
1030   <sequence>  
1031     <element ref="saml:Issuer" minOccurs="0"/>  
1032     <element ref="ds:Signature" minOccurs="0"/>  
1033     <element ref="samlp:RelayState" minOccurs="0"/>  
1034     <element ref="samlp:Extensions" minOccurs="0"/>  
1035   </sequence>  
1036   <attribute name="RequestID" type="ID" use="required"/>  
1037   <attribute name="MajorVersion" type="integer" use="required"/>  
1038   <attribute name="MinorVersion" type="integer" use="required"/>  
1039   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1040   <attribute name="Consent" type="anyURI" use="optional"/>  
1041 </complexType>
```

```
1042 <element name="Extensions" type="samlp:ExtensionsType"/>
1043 <complexType name="ExtensionsType">
1044   <sequence>
1045     <any namespace="##any" processContents="lax"
1046     maxOccurs="unbounded"/>
1047   </sequence>
1048 </complexType>
```

### 1049 3.2.1.1 Element <RelayState>

1050 SAML requests MAY contain a string-valued element containing state information that the requester  
1051 wishes the responder to include in the response. This is particularly useful with asynchronous bindings  
1052 of protocol messages, such as the encoding of messages in browser URLs. This data SHOULD be  
1053 integrity-protected by the requester and MAY have other protections placed on it by the requester, such  
1054 as confidentiality. The length of this value SHOULD be kept as short as possible because of limitations  
1055 of the bindings in which it may be needed.

1056 The following schema fragment defines the <RelayState> element:

```
1057 <element name="RelayState" type="string"/>
```

## 1058 3.2.2 Complex Type StatusResponseType

1059 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This  
1060 type defines common attributes and elements that are associated with all SAML responses:

### 1061 ResponseID [Required]

1062 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in  
1063 Section 1.2.3 for identifier uniqueness.

### 1064 InResponseTo [Optional]

1065 A reference to the identifier of the request to which the response corresponds, if any. If the response  
1066 is not generated in response to a request, or if the `RequestID` attribute value of a request cannot be  
1067 determined (because the request is malformed), then this attribute MUST NOT be present.  
1068 Otherwise, it MUST be present and its value MUST match the value of the corresponding  
1069 `RequestID` attribute value.

### 1070 MajorVersion [Required]

1071 The major version of this response. The identifier for the version of SAML defined in this  
1072 specification is 2. SAML versioning is discussed in Section SAML Versioning.

### 1073 MinorVersion [Required]

1074 The minor version of this response. The identifier for the version of SAML defined in this  
1075 specification is 0. SAML versioning is discussed in Section SAML Versioning.

### 1076 IssueInstant [Required]

1077 The time instant of issue of the response. The time value is encoded in UTC as described in Section  
1078 Time Values.

### 1079 Recipient [Optional]

1080 The intended recipient of this response. This is useful to prevent malicious forwarding of responses  
1081 to unintended recipients, a protection that is required by some use profiles. It is set by the generator  
1082 of the response to a URI reference that identifies the intended recipient. If present, the actual  
1083 recipient MUST check that the URI reference identifies the recipient or a resource managed by the  
1084 recipient. If it does not, the response MUST be discarded.

- 1085 <Issuer> [Optional]  
 1086 Identifies the entity that generated the response message.
- 1087 <ds:Signature> [Optional]  
 1088 An XML Signature that authenticates the response, as described in Section SAML and XML  
 1089 Signature Syntax and Processing.
- 1090 <RelayState> [Optional]  
 1091 This contains state information from the associated request being relayed back in the response. It  
 1092 MUST match the <RelayState> value in the associated request, if any.
- 1093 <Extensions> [Optional]  
 1094 This contains optional protocol message extension elements that are agreed upon between the  
 1095 communicating parties.
- 1096 <Status> [Required]  
 1097 A code representing the status of the corresponding request.

1098 The following schema fragment defines the **StatusResponseType** complex type:

```

1099 <complexType name="StatusResponseType">
1100   <sequence>
1101     <element ref="saml:Issuer" minOccurs="0"/>
1102     <element ref="ds:Signature" minOccurs="0"/>
1103     <element ref="samlp:RelayState" minOccurs="0"/>
1104     <element ref="samlp:Extensions" minOccurs="0"/>
1105     <element ref="samlp:Status"/>
1106   </sequence>
1107   <attribute name="ResponseID" type="ID" use="required"/>
1108   <attribute name="InResponseTo" type="NCName" use="optional"/>
1109   <attribute name="MajorVersion" type="integer" use="required"/>
1110   <attribute name="MinorVersion" type="integer" use="required"/>
1111   <attribute name="IssueInstant" type="dateTime" use="required"/>
1112   <attribute name="Recipient" type="anyURI" use="optional"/>
1113 </complexType>

```

### 1114 3.2.2.1 Element <Status>

1115 The <Status> element contains the following elements:

- 1116 <StatusCode> [Required]  
 1117 A code representing the status of the corresponding request.
- 1118 <StatusMessage> [Optional]  
 1119 A message which MAY be returned to an operator.
- 1120 <StatusDetail> [Optional]  
 1121 Additional information concerning an error condition.

1122 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1123 <element name="Status" type="samlp:StatusType"/>
1124 <complexType name="StatusType">
1125   <sequence>
1126     <element ref="samlp:StatusCode"/>
1127     <element ref="samlp:StatusMessage" minOccurs="0"/>
1128     <element ref="samlp:StatusDetail" minOccurs="0"/>
1129   </sequence>
1130 </complexType>

```

### 1131 **3.2.2.2 Element <StatusCode>**

1132 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the  
1133 corresponding request. The <StatusCode> element has the following element and attribute:

1134 Value [Required]

1135 The status code value. This attribute contains an XML Schema QName; a namespace prefix MUST  
1136 be provided. The value of the topmost <StatusCode> element MUST be from the top-level list  
1137 provided in this section.

1138 <StatusCode> [Optional]

1139 A subordinate status code that provides more specific information on an error condition.

1140 The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The  
1141 local parts of these QNames are as follows:

1142 Success

1143 The request succeeded.

1144 VersionMismatch

1145 The SAML responder could not process the request because the version of the request message  
1146 was incorrect.

1147 Requester

1148 The request could not be performed due to an error on the part of the requester.

1149 Responder

1150 The request could not be performed due to an error on the part of the SAML responder or SAML  
1151 authority.

1152 The following second-level status codes are referenced at various places in the specification. Additional  
1153 second-level status codes MAY be defined in future versions of the SAML specification.

1154 RequestVersionTooHigh

1155 The SAML responder cannot process the request because the protocol version specified in the  
1156 request message is a major upgrade from the highest protocol version supported by the responder.

1157 RequestVersionTooLow

1158 The SAML responder cannot process the request because the protocol version specified in the  
1159 request message is too low.

1160 RequestVersionDeprecated

1161 The SAML responder can not process any requests with the protocol version specified in the  
1162 request.

1163 TooManyResponses

1164 The response message would contain more elements than the SAML responder will return.

1165 RequestDenied

1166 The SAML responder or SAML authority is able to process the request but has chosen not to  
1167 respond. This status code MAY be used when there is concern about the security context of the  
1168 request message or the sequence of request messages received from a particular requester.

1169 ResourceNotRecognized

1170 The SAML authority does not wish to support resource-specific attribute queries, or the resource  
1171 value provided in the request message is invalid or unrecognized.

1172 FederationDoesNotExist

1173 The responding provider does not recognize the federated <NameIdentifier> in the request.

1174 SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT  
1175 define additional codes in the SAML assertion or protocol namespace.

1176 The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value  
1177 attribute and have the above semantics only in that context.

1178 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex  
1179 type:

```
1180 <element name="StatusCode" type="samlp:StatusCodeType"/>
1181 <complexType name="StatusCodeType">
1182   <sequence>
1183     <element ref="samlp:StatusCode" minOccurs="0"/>
1184   </sequence>
1185   <attribute name="Value" type="QName" use="required"/>
1186 </complexType>
```

### 1187 3.2.2.3 Element <StatusMessage>

1188 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1189 The following schema fragment defines the <StatusMessage> element:

```
1190 <element name="StatusMessage" type="string"/>
```

### 1191 3.2.2.4 Element <StatusDetail>

1192 The <StatusDetail> element MAY be used to specify additional information concerning an error  
1193 condition.

1194 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
1195 complex type:

```
1196 <element name="StatusDetail" type="samlp:StatusDetailType"/>
1197 <complexType name="StatusDetailType">
1198   <sequence>
1199     <any namespace="##any" processContents="lax" minOccurs="0"
1200     maxOccurs="unbounded"/>
1201   </sequence>
1202 </complexType>
```

## 1203 3.3 Assertion Query and Request Protocol

1204 This section defines messages and processing rules for requesting existing assertions by reference,  
1205 artifact, or querying for assertions by subject and statement type.

### 1206 3.3.1 Element <AssertionIDRequest>

1207 If the requester knows the AssertionID of one or more assertions, the <AssertionIDRequest>  
1208 message can be used to request that the assertion(s) be returned in a <Response> message. The  
1209 <saml:AssertionIDReference> element is used to specify the assertion(s) to return. See Section  
1210 Element <AssertionIDReference> for more information on this element.

1211 The following schema fragment defines the <AssertionIDRequest> element:

```

1212 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
1213 <complexType name="AssertionIDRequestType">
1214   <complexContent>
1215     <extension base="samlp:RequestAbstractType">
1216       <sequence>
1217         <element ref="saml:AssertionIDReference"
1218 maxOccurs="unbounded"/>
1219       </sequence>
1220     </extension>
1221   </complexContent>
1222 </complexType>

```

### 1223 3.3.2 Element <ArtifactRequest>

1224 The <ArtifactRequest> message is used to request that one or more assertions be returned in a  
1225 <Response> message by specifying one or more assertion artifacts that represent the assertion(s)  
1226 being requested. Its use is governed by the specific profile of SAML that is being used; see the SAML  
1227 specification for bindings and profiles [SAMLBind] for more information on the use of assertion artifacts  
1228 in profiles.

1229 The following schema fragment defines the <ArtifactRequest> element:

```

1230 <element name="ArtifactRequest" type="samlp:ArtifactRequestType"/>
1231 <complexType name="ArtifactRequestType">
1232   <complexContent>
1233     <extension base="samlp:RequestAbstractType">
1234       <sequence>
1235         <element ref="samlp:AssertionArtifact"
1236 maxOccurs="unbounded"/>
1237       </sequence>
1238     </extension>
1239   </complexContent>
1240 </complexType>
1241 <element name="AssertionArtifact" type="string"/>

```

### 1242 3.3.3 Queries

1243 The following sections define the SAML query request messages.

#### 1244 3.3.3.1 Element <SubjectQuery>

1245 The <SubjectQuery> message element is an extension point that allows new SAML queries to be  
1246 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and  
1247 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the <Subject>  
1248 element, and an optional SessionIndex attribute to **RequestAbstractType**.

1249 SessionIndex [Optional]

1250 **If present, specifies a filter for possible responses. Such a query asks the question “What**  
1251 **assertions containing subject statements do you have for this subject within the context of**  
1252 **the supplied session information?”**

1253 If the SessionIndex attribute is present in any defined query, at least one element that extends  
1254 **SubjectStatementAbstractType** in the set of returned assertions MUST contain an SessionIndex  
1255 attribute that matches the SessionIndex attribute in the query. It is OPTIONAL for the  
1256 complete set of all such matching assertions to be returned in the response.

1257 The following schema fragment defines the <SubjectQuery> element and its  
1258 **SubjectQueryAbstractType** complex type:

```

1259 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
1260 <complexType name="SubjectQueryAbstractType" abstract="true">
1261   <complexContent>
1262     <extension base="samlp:RequestAbstractType">
1263       <sequence>
1264         <element ref="saml:Subject"/>
1265       </sequence>
1266       <attribute name="SessionIndex" type="string"
1267 use="optional"/>
1268     </extension>
1269   </complexContent>
1270 </complexType>

```

### 1271 3.3.3.2 Element <AuthenticationQuery>

1272 The <AuthenticationQuery> message element is used to make the query “What assertions  
1273 containing authentication statements are available for this subject?” A successful <Response> will  
1274 contain one or more assertions containing authentication statements.

1275 The <AuthenticationQuery> message MUST NOT be used as a request for a new authentication  
1276 using credentials provided in the request. <AuthenticationQuery> is a request for statements about  
1277 authentication acts that have occurred in a previous interaction between the indicated subject and the  
1278 Authentication Authority.

1279 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the  
1280 addition of the following attribute:

1281 AuthenticationMethod [Optional]

1282 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1283 containing authentication statements do you have for this subject with the supplied authentication  
1284 method?”

1285 In response to an authentication query, a SAML authority returns assertions with authentication  
1286 statements as follows:

- 1287 • Rules given in Section Responses to for matching against the <Subject> element of the query  
1288 identify the assertions that may be returned.
- 1289 • If the AuthenticationMethod attribute is present in the query, at least one  
1290 <AuthenticationStatement> element in the set of returned assertions MUST contain an  
1291 AuthenticationMethod attribute that matches the AuthenticationMethod attribute in  
1292 the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in  
1293 the response.

1294 The following schema fragment defines the <AuthenticationQuery> element and its  
1295 **AuthenticationQueryType** complex type:

```

1296 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
1297 <complexType name="AuthenticationQueryType">
1298   <complexContent>
1299     <extension base="samlp:SubjectQueryAbstractType">
1300       <attribute name="AuthenticationMethod" type="anyURI"/>
1301     </extension>
1302   </complexContent>
1303 </complexType>

```



### 1304 3.3.3.3 Element <AttributeQuery>

1305 The <AttributeQuery> element is used to make the query "Return the requested attributes for this  
1306 subject." A successful response will be in the form of assertions containing attribute statements. This  
1307 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of  
1308 the following element and attribute:

1309 Resource [Optional]

1310 If present, specifies that the attribute query is being made in order to evaluate a specific access  
1311 request relating to the resource. The SAML authority MAY use the resource attribute to establish the  
1312 scope of the request. It is permitted for this attribute to have the value of the empty URI reference  
1313 (""), and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]  
1314 §4.2.

1315 If the resource attribute is specified and the SAML authority does not wish to support resource-  
1316 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the  
1317 Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a  
1318 second-level <StatusCode> value of ResourceNotRecognized.

1319 <AttributeDesignator> [Any Number] (see Section Time Values)

1320 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no  
1321 attributes are specified, it indicates that all attributes allowed by policy are requested.

1322 In response to an attribute query, a SAML authority returns assertions with attribute statements as  
1323 follows:

- 1324 • Rules given in Section Responses to for matching against the <Subject> element of the query  
1325 identify the assertions that may be returned.
- 1326 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute  
1327 values returned, as noted above.
- 1328 • The SAML authority MAY take the Resource attribute into account in further constraining the values  
1329 returned, as noted above.
- 1330 • The attribute values returned MAY be constrained by application-specific policy considerations.

1331 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**  
1332 complex type:

```
1333 <element name="AttributeQuery" type="saml:AttributeQueryType"/>  
1334 <complexType name="AttributeQueryType">  
1335   <complexContent>  
1336     <extension base="saml:SubjectQueryAbstractType">  
1337       <sequence>  
1338         <element ref="saml:AttributeDesignator"  
1339           minOccurs="0" maxOccurs="unbounded"/>  
1340       </sequence>  
1341       <attribute name="Resource" type="anyURI" use="optional"/>  
1342     </extension>  
1343   </complexContent>  
1344 </complexType>
```

### 1345 3.3.3.4 Element <AuthorizationDecisionQuery>

1346 The <AuthorizationDecisionQuery> element is used to make the query "Should these actions on  
1347 this resource be allowed for this subject, given this evidence?" A successful response will be in the form  
1348 of assertions containing authorization decision statements. This element is of type

1349 **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the  
1350 following elements and attribute:

1351 Resource [Required]

1352 A URI reference indicating the resource for which authorization is requested.

1353 <Action> [One or More]

1354 The actions for which authorization is requested.

1355 <Evidence> [Optional]

1356 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1357 In response to an authorization decision query, a SAML authority returns assertions with authorization  
1358 decision statements as follows:

- 1359 • Rules given in Section Responses to for matching against the <Subject> element of the query  
1360 identify the assertions that may be returned.

1361 The following schema fragment defines the <AuthorizationDecisionQuery> element and its  
1362 **AuthorizationDecisionQueryType** complex type:

```
1363 <element name="AuthorizationDecisionQuery"  
1364 type="samlp:AuthorizationDecisionQueryType"/>  
1365 <complexType name="AuthorizationDecisionQueryType">  
1366 <complexContent>  
1367 <extension base="samlp:SubjectQueryAbstractType">  
1368 <sequence>  
1369 <element ref="saml:Action" maxOccurs="unbounded"/>  
1370 <element ref="saml:Evidence" minOccurs="0"/>  
1371 </sequence>  
1372 <attribute name="Resource" type="anyURI" use="required"/>  
1373 </extension>  
1374 </complexContent>  
1375 </complexType>
```

### 1376 3.3.4 Element <Response>

1377 The <Response> message element is used when a response consists of a list of zero or more  
1378 assertions that answer the request. It has the complex type **ResponseType**, which extends  
1379 **StatusResponseType** by adding the following element:

1380 <Assertion> [Any Number]

1381 Specifies an assertion by value. (See Section Element <Assertion> for more information.)

1382 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1383 <element name="Response" type="samlp:ResponseType"/>  
1384 <complexType name="ResponseType">  
1385 <complexContent>  
1386 <extension base="samlp:StatusResponseType">  
1387 <sequence>  
1388 <element ref="saml:Assertion" minOccurs="0"  
1389 maxOccurs="unbounded"/>  
1390 </sequence>  
1391 </extension>  
1392 </complexContent>  
1393 </complexType>
```

### 1394 3.3.4.1 Responses to Queries

1395 In response to a query message, every assertion returned by a SAML authority **MUST** contain at least  
1396 one statement whose `<saml:Subject>` element **strongly matches** the `<saml:Subject>` element  
1397 found in the query.

1398 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both  
1399 apply:

- 1400 • If S2 includes a name identifier element (any element whose type is derived from  
1401 **NameIdentifierAbstractType**), then S1 must include an identical name identifier element.
- 1402 • If S2 includes a `<saml:SubjectConfirmation>` element, then S1 must include an identical  
1403 `<saml:SubjectConfirmation>` element.

1404 If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
1405 expressed by a query, the `<Response>` element **MUST NOT** contain an `<Assertion>` element and  
1406 **MUST** include a `<StatusCode>` element with value `Success`. It **MAY** return a `<StatusMessage>`  
1407 element with additional information.

## 1408 3.4 Federated Name Registration Protocol

1409 When an identity provider and service provider first federate a principal's identity using a  
1410 `<NameIdentifier>` element with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1411 `format:federated`, the identity provider generates an opaque value that serves as the initial name  
1412 identifier that both the service provider and the identity provider use in referring to the principal when  
1413 communicating with each other.

1414 Subsequent to federation, the service provider **MAY** register a different opaque value with the identity  
1415 provider. This opaque value is an attribute termed the `SPProvidedIdentifier`. Until the service provider  
1416 registers a different name, this attribute is omitted from `<NameIdentifier>` elements referring to the  
1417 principal.

1418 Either the service provider or the identity provider **MAY** register a new name identifier for a principal with  
1419 each other at any time following federation. The name identifiers specified by providers **SHOULD** be  
1420 unique across the identity providers with which the principal's identity is federated and **SHOULD** be  
1421 unique within the group of name identifiers that have been registered with the identity provider by this  
1422 service provider.

1423 Only federated identifiers (as defined by a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1424 `format:federated`) can be replaced and set with this protocol; non-federated, encrypted, or transient  
1425 identifiers **MUST NOT** be used.

### 1426 3.4.1 Element `<RegisterNameIdentifierRequest>`

1427 To register an `SPProvidedIdentifier` attribute with an identity provider, the service provider sends a  
1428 `<RegisterNameIdentifierRequest>` message. The same message may be sent by an identity  
1429 provider, seeking to change the `<NameIdentifier>` value stored by the service provider.

1430 The `<RegisterNameIdentifierRequest>` message **SHOULD** be signed or otherwise authenticated  
1431 and integrity protected.

1432 The `<Issuer>` of the request **MUST** contain the unique identifier of the requesting provider, with a  
1433 `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

1434 This message has the complex type **RegisterNameIdentifierRequestType**, which extends  
1435 **RequestAbstractType** and adds the following elements:

1436 <NameIdentifier> [Required]

1437 The federated name identifier and associated attributes that specify the principal as currently  
1438 recognized by the identity and service providers prior to this request.

1439 <NewIdentifier> [Required]

1440 The new federated identifier value to be used when communicating with the requesting provider  
1441 concerning this principal. If the requester is the service provider, the new identifier will appear in  
1442 subsequent <NameIdentifier> elements in the *SPProvidedIdentifier* attribute. If the  
1443 requester is the identity provider, the new value will appear in subsequent <NameIdentifier>  
1444 elements as the element's value.

1445 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its  
1446 **RegisterNameIdentifierRequestType** complex type:

```
1447 <element name="NewIdentifier" type="string">  
1448 <element name="RegisterNameIdentifierRequest"  
1449 type="samlp:RegisterNameIdentifierRequestType"/>  
1450 <complexType name="RegisterNameIdentifierRequestType">  
1451 <complexContent>  
1452 <extension base="samlp:RequestAbstractType">  
1453 <sequence>  
1454 <element ref="saml:NameIdentifier"/>  
1455 <element ref="samlp:NewIdentifier"/>  
1456 </sequence>  
1457 </extension>  
1458 </complexContent>  
1459 </complexType>
```

### 1460 3.4.2 Element <RegisterNameIdentifierResponse>

1461 The recipient of a <RegisterNameIdentifierRequest> message MUST respond with a  
1462 <RegisterNameIdentifierResponse> message, which is of type **StatusResponseType** with no  
1463 additional content.

1464 The <RegisterNameIdentifierResponse> message SHOULD be signed or otherwise authenticated  
1465 and integrity protected.

1466 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
1467 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

1468 The following schema fragment defines the <RegisterNameIdentifierResponse> element:

```
1469 <element name="RegisterNameIdentifierResponse"  
1470 type="samlp:StatusResponseType"/>
```

### 1471 3.4.3 Processing Rules

1472 The recipient MUST validate any signature present on the request or response message. To be  
1473 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
1474 message.

1475 If the request includes a <NameIdentifier> for which no federation exists between the service  
1476 provider and the identity provider, the responding provider MUST respond with a <samlp:Status>  
1477 containing a second-level <samlp:StatusCode> of *samlp:FederationDoesNotExist*.

1478 If the service provider requests that its identifier be changed, the identity provider MUST include the  
1479 <NewIdentifier> element's value as the `SPProvidedIdentifier` when subsequently  
1480 communicating to the service provider regarding this principal.

1481 If the identity provider requests that its identifier be changed, the service provider MUST use the  
1482 <NewIdentifier> element's value as the <NameIdentifier> element value when subsequently  
1483 communicating with the identity provider regarding this principal.

1484 In either case, the <NameIdentifier> value in the request and its associated  
1485 `SPProvidedIdentifier` attribute MUST contain the most recent name identifier information  
1486 established between the providers for the principal. The `NameQualifier` attribute MUST contain the  
1487 unique identifier of the identity provider. If the principal's identity federation is between the identity  
1488 provider and an affiliation group of which the service provider is a member, then the `SPNameQualifier`  
1489 attribute MUST contain the unique identifier of the affiliation group. Otherwise, it MUST contain the  
1490 unique identifier of the service provider.

1491 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
1492 systems at both the requester and the responder. Implementations might wish to allow each party to  
1493 accept either identifier for some period of time following the successful completion of a name identifier  
1494 change. Not doing so could result in the inability of the principal to access resources.

1495 All other processing rules associated with the underlying request and response messages MUST be  
1496 observed.

## 1497 **3.5 Federation Termination Protocol**

1498 When a principal (or an appropriate agent acting on his or her behalf) terminates an identity federation  
1499 between a service provider and an identity provider through an interaction with the service provider, the  
1500 service provider MUST send a <FederationTerminationNotification> message to the identity  
1501 provider. The service provider is stating that it will no longer accept authentication assertions from the  
1502 identity provider for the specified principal.

1503 Likewise, when a principal terminates an identity federation through an interaction with the identity  
1504 provider, the identity provider MUST send a <FederationTerminationNotification> message to  
1505 the service provider. In this case, the identity provider is stating that it will no longer provide  
1506 authentication assertions to the service provider for the specified principal.

1507 Only federated identifiers (as defined by a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1508 `format:federated`) can be replaced and set with this protocol; non-federated, encrypted, or transient  
1509 identifiers MUST NOT be used.

### 1510 **3.5.1 Element <FederationTerminationNotification>**

1511 A provider sends a <FederationTerminationNotification> to the provider with which it is  
1512 terminating a federation. The <FederationTerminationNotification> message SHOULD be  
1513 signed or otherwise authenticated and integrity protected.

1514 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
1515 `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

1516 This message has the complex type **FederationTerminationNotificationType**, which extends  
1517 **RequestAbstractType** and adds the following elements:

1518 <NameIdentifier> [Required]

1519     The federated name identifier and associated attributes that specify the principal as currently  
1520     recognized by the identity and service providers prior to this request. `Format` MUST be

1521 urn:oasis:names:tc:SAML:2.0:nameid-format:federated.

1522 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its  
1523 RegisterNameIdentifierRequestType complex type:

```
1524 <element name="FederationTerminationNotification"  
1525 type="saml:FederationTerminationNotificationType"/>  
1526 <complexType name="FederationTerminationNotificationType">  
1527 <complexContent>  
1528 <extension base="saml:RequestAbstractType">  
1529 <sequence>  
1530 <element ref="saml:NameIdentifier"/>  
1531 </sequence>  
1532 </extension>  
1533 </complexContent>  
1534 </complexType>
```

### 1535 3.5.2 Element <FederationTerminationResponse>

1536 The recipient of a <FederationTerminationNotification> message MUST respond with a  
1537 <FederationTerminationResponse> message, which is of type **StatusResponseType** with no  
1538 additional content.

1539 The <FederationTerminationResponse> message SHOULD be signed or otherwise authenticated  
1540 and integrity protected.

1541 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
1542 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

1543 The following schema fragment defines the <FederationTerminationResponse> element:

```
1544 <element name="FederationTerminationResponse"  
1545 type="saml>StatusResponseType"/>
```

### 1546 3.5.3 Processing Rules

1547 The recipient MUST validate any signature present on the request or response message. To be  
1548 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
1549 message.

1550 If the request includes a <NameIdentifier> for which no federation exists between the service  
1551 provider and the identity provider, the responding provider MUST respond with a <saml:Status>  
1552 containing a second-level <saml:StatusCode> of saml:FederationDoesNotExist.

1553 Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been  
1554 terminated. A provider MAY choose to invalidate the session of a user for whom federation has been  
1555 terminated.

## 1556 3.6 Single Logout Protocol

1557  
1558 The single logout protocol provides a message exchange protocol by which all sessions provided by a  
1559 particular session authority are near-simultaneously terminated. The single logout protocol is used either  
1560 when a principal logs out at a session participant or when the principal logs out directly at the  
1561 session authority. This protocol may also be used to logout a principal due to a timeout. The reason for  
1562 the logout event may be indicated through the `reason` attribute.

1563  
1564 The principal may have established authenticated sessions both with the session authority, and  
1565 individual session participants, based on authentication assertions supplied by the session authority.

1566  
1567 When the Principal invokes the single logout process at a Session participant, the session participant  
1568 MUST send a `<LogoutRequest>` message to the session authority that provided the authentication  
1569 service related to that session at the session participant.

1570  
1571 When either the principal invokes a logout at the session authority, or a session participant sends a  
1572 logout request to the session authority specifying that principal, the session authority MUST send a  
1573 `<LogoutRequest>` message to each session participant to which it provided authentication assertions  
1574 under its current session with the principal, with the exception of the session participant that sent the  
1575 `<LogoutRequest>` message to the session authority.

### 1577 3.6.1 Element `<LogoutRequest>`

1578 `<NameIdentifier>` [Required]

1579 The federated name identifier and associated attributes that specify the principal as currently  
1580 recognized by the identity and service providers prior to this request.

1581 `<SessionIndex>` [Optional]

1582 The identifier that indexes this session at the message recipient.

1583 `NotOnOrAfter` [Optional]

1584 The time at which the request expires.

1585 `reason` [Optional]

1586 An indication of the reason for the logout.

1587

```
1588 <element name="LogoutRequest" type="saml:LogoutRequestType"/>
1589 <complexType name="LogoutRequestType">
1590   <complexContent>
1591     <extension base="saml:RequestAbstractType">
1592       <sequence>
1593         <element ref="saml:NameIdentifier"/>
1594         <element name="SessionIndex" type="string" minOccurs="0"
1595 maxOccurs="unbounded"/>
1596       </sequence>
1597       <attribute name="reason" type="string" minOccurs="0"/>
1598       <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
1599     </extension>
1600   </complexContent>
1601 </complexType>
```

1602 **3.6.2 Element <LogoutResponse> (UNFINISHED)**

1603 `<element name="LogoutResponse" type="samlp:StatusResponseType"/>`

1604 **3.6.3 Processing Rules (UNFINISHED)**

1605 The <Issuer> of either message in this protocol MUST contain the unique identifier of the requesting  
1606 provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

1607 Message recipients MUST validate any signature present on the messages specified in this protocol. To  
1608 be considered valid, the signature provided must be the signature of the <Issuer> contained in the  
1609 message.



---

## 1610 4 SAML Versioning

1611 The SAML specification set is versioned in two independent ways. Each is discussed in the following  
1612 sections, along with processing rules for detecting and handling version differences, when applicable.  
1613 Also included are guidelines on when and why specific version information is expected to change in  
1614 future revisions of the specification.

1615 When version information is expressed as both a Major and Minor version, it may be expressed  
1616 discretely, or in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version  
1617 number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

1618  $Major_B > Major_A \vee ( ( Major_B = Major_A ) \wedge Minor_B > Minor_A )$

### 1619 4.1 SAML Specification Set Version

1620 Each release of the SAML specification set will contain a major and minor version designation describing  
1621 its relationship to earlier and later versions of the specification set. The version will be expressed in the  
1622 content and filenames of published materials, including the specification set document(s), and XML  
1623 schema instance(s). There are no normative processing rules surrounding specification set versioning,  
1624 since it merely encompasses the collective release of normative specification documents which  
1625 themselves contain processing rules.

1626 The overall size and scope of changes to the specification set document(s) will informally dictate whether  
1627 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards  
1628 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics  
1629 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a  
1630 major revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0,  
1631 described in Section Interoperability with SAML V1.0.

#### 1632 4.1.1 Schema Version

1633 As a non-normative documentation mechanism, any XML schema instances published as part of the  
1634 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the  
1635 specification set version in which it has been published. Validating implementations MAY use the  
1636 attribute as a means of distinguishing which version of a schema is being used to validate messages, or  
1637 to support a multiplicity of versions of the same logical schema.

#### 1638 4.1.2 SAML Assertion Version

1639 The SAML <Assertion> element contains attributes for expressing the major and minor version of the  
1640 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to  
1641 document the syntax, semantics, and processing rules of the assertions of the same version. That is,  
1642 specification set version 1.0 describes assertion version 1.0, and so on.

1643 There is explicitly NO relationship between the assertion version and the SAML assertion XML  
1644 namespace that contains the schema definitions for that assertion version.

1645 The following processing rules apply:

- 1646 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported  
1647 by the authority.
- 1648 • A SAML relying party MUST NOT process any assertion with a major assertion version number not  
1649 supported by the relying party.

- 1650 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version  
1651 number is higher than the minor assertion version number supported by the relying party. However,  
1652 all assertions that share a major assertion version number MUST share the same general processing  
1653 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1  
1654 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0  
1655 assertion without ill effect.

### 1656 4.1.3 SAML Protocol Version

1657 The SAML protocol <Request> and <Response> elements contain attributes for expressing the major  
1658 and minor version of the request or response message using a pair of integers. Each version of the  
1659 SAML specification set will be construed so as to document the syntax, semantics, and processing rules  
1660 of the protocol messages of the same version. That is, specification set version 1.0 describes request  
1661 and response version V1.0, and so on.

1662 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace  
1663 that contains the schema definitions for protocol messages for that protocol version.

1664 The version numbers used in SAML protocol <Request> and <Response> elements will be the same  
1665 for any particular revision of the SAML specification set.

#### 1666 4.1.3.1 Request Version

1667 The following processing rules apply to requests:

- 1668 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
1669 SAML requester and the SAML responder.
- 1670 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume  
1671 that it supports requests with the highest request version supported by the requester.
- 1672 • A SAML requester MUST NOT issue a request message with a request version number matching a  
1673 response version number that the requester does not support.
- 1674 • A SAML responder MUST reject any request with a major request version number not supported by  
1675 the responder.
- 1676 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
1677 higher than the highest supported request version that it supports. However, all requests that share a  
1678 major request version number MUST share the same general processing rules and semantics, and  
1679 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax  
1680 of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

### 1681 4.1.4 Response Version

1682 The following processing rules apply to responses:

- 1683 • A SAML responder MUST NOT issue a response message with a response version number higher  
1684 than the request version number of the corresponding request message.
- 1685 • A SAML responder MUST NOT issue a response message with a major response version number  
1686 lower than the major request version number of the corresponding request message except to report  
1687 the error `RequestVersionTooHigh`.

1688 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-  
1689 level <StatusCode> value of `VersionMismatch`, and MAY result in reporting one of the following

1690 second-level values: RequestVersionTooHigh, RequestVersionTooLow, or  
1691 RequestVersionDeprecated.

## 1692 4.1.5 Permissible Version Combinations

1693 In general, assertions of a particular major version may appear in response messages of the same major  
1694 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol  
1695 schema. Future versions of this specification are expected to explicitly describe the permitted  
1696 combinations across major versions.

1697 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to  
1698 appear in a V1.1 response message.

## 1699 4.2 SAML Namespace Version

1700 XML schema instances and "qualified names" (QNames) published as part of the specification set  
1701 contain one or more target namespaces into which the type, element, and attribute definitions are  
1702 placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and  
1703 syntactical definitions that make up that part of the specification.

1704 The namespace URIs defined by the specification set will generally contain version information of the  
1705 form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to  
1706 the major and minor version of the specification set in which the namespace is first introduced and  
1707 defined. This information is not typically consumed by an XML processor, which treats the namespace  
1708 opaquely, but is intended to communicate the relationship between the specification set and the  
1709 namespaces it defines.

1710 As a general rule, implementers can expect the namespaces (and the associated schema definitions)  
1711 defined by a major revision of the specification set to remain valid and stable across minor revisions of  
1712 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,  
1713 but this is expected to be rare. In such cases, the older namespaces and their associated definitions  
1714 should be expected to remain valid until a major specification set revision.

### 1715 4.2.1 Schema Evolution

1716 In general, maintaining namespace stability while adding or changing the content of a schema are  
1717 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
1718 older implementations will react to any given change, making forward compatibility difficult to achieve.  
1719 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of  
1720 namespace stability. Except in special circumstances (for example to correct major deficiencies or fix  
1721 errors), implementations should expect forward compatible schema changes in minor revisions, allowing  
1722 new messages to validate against older schemas.

1723 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
1724 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
1725 that leverage the extension facilities described in Section SAML Extensions. Older implementations  
1726 SHOULD reject such extensions gracefully when they are encountered in contexts that dictate mandatory  
1727 semantics. Examples include new query, statement, or condition types.

1728

## 5 SAML and XML Signature Syntax and Processing

1729 SAML assertions and SAML protocol request and response messages may be signed, with the following  
1730 benefits:

- 1731 • An assertion signed by the SAML authority supports:
  - 1732 – Assertion integrity.
  - 1733 – Authentication of the SAML authority to a SAML relying party.
  - 1734 – If the signature is based on the SAML authority's public-private key pair, then it also provides for  
1735 non-repudiation of origin.
- 1736 • A SAML protocol request or response message signed by the message originator supports:
  - 1737 – Message integrity.
  - 1738 – Authentication of message origin to a destination.
  - 1739 – If the signature is based on the originator's public-private key pair, then it also provides for non-  
1740 repudiation of origin.

1741 A digital signature is not always required in SAML. For example, it may not be required in the following  
1742 situations:

- 1743 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains  
1744 protection from a signature on the containing protocol response message. "Inherited" signatures  
1745 should be used with care when the contained object (such as the assertion) is intended to have a  
1746 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,  
1747 exposing the XML content and adding potentially unnecessary overhead.
- 1748 • The SAML relying party or SAML requester may have obtained an assertion or protocol message  
1749 from the SAML authority or SAML responder directly (with no intermediaries) through a secure  
1750 channel, with the SAML authority or SAML responder having authenticated to the relying party or  
1751 SAML responder by some means other than a digital signature.

1752 Many different techniques are available for "direct" authentication and secure channel establishment  
1753 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1754 the applicable security requirements depend on the communicating applications and the nature of the  
1755 assertion or message transported.

1756 It is recommended that, in all other contexts, digital signatures be used for assertions and request and  
1757 response messages. Specifically:

- 1758 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority  
1759 SHOULD be signed by the SAML authority.
- 1760 • A SAML protocol message arriving at a destination from an entity other than the originating site  
1761 SHOULD be signed by the origin site.

1762 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that  
1763 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures  
1764 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure  
1765 compatibility with profiles that may require other mechanisms.

1766 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST  
1767 be used if signing.

## 1768 **5.1 Signing Assertions**

1769 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema  
1770 as described in Section Assertions.

## 1771 **5.2 Request/Response Signing**

1772 All SAML protocol request and response messages MAY be signed using the XML Signature. This is  
1773 reflected in the schema as described in Sections Requests and Responses and .

## 1774 **5.3 Signature Inheritance**

1775 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`  
1776 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a  
1777 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a  
1778 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,  
1779 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
1780 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
1781 and signature options.

1782 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
1783 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may  
1784 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
1785 information from the surrounding context, but no such inheritance should be inferred unless specifically  
1786 identified by the profile.

## 1787 **5.4 XML Signature Profile**

1788 The XML Signature specification calls out a general XML syntax for signing data with flexibility and  
1789 many choices. This section details the constraints on these facilities so that SAML processors do not  
1790 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
1791 `xsd:ID`-typed attributes optionally present on the root elements to which signatures can apply: the  
1792 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the  
1793 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section  
1794 as the identifier attributes.

### 1795 **5.4.1 Signing Formats and Algorithms**

1796 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1797 detached.

1798 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
1799 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
1800 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 1801 **5.4.2 References**

1802 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the  
1803 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element  
1804 may or may not be the root element of the actual XML document containing the signed assertion or  
1805 message.

1806 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier  
1807 attribute value of the root element of the message being signed. For example, if the attribute value is  
1808 "foo", then the URI attribute in the `<ds:Reference>` element MUST be "#foo".

### 1809 **5.4.3 Canonicalization Method**

1810 SAML implementations SHOULD use Exclusive Canonicalization , with or without comments, both in the  
1811 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`  
1812 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages  
1813 embedded in an XML context can be verified independent of that context.

### 1814 **5.4.4 Transforms**

1815 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
1816 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive  
1817 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1818 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1819 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1820 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This  
1821 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
1822 applying the transforms manually to the content and reverifying the result as consisting of the same  
1823 SAML message.

### 1824 **5.4.5 KeyInfo**

1825 XML Signature defines usage of the `<ds:KeyInfo>` element . SAML does not require the use of  
1826 `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be  
1827 absent.

### 1828 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

1829 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section  
1830 SAML Assertions.

### 1831 **5.4.7 Interoperability with SAML V1.0**

1832 The use of XML Signature described above is incompatible with the usage described in the SAML V1.0  
1833 specification [SAMLCore1.0]. The original profile was underspecified and was insufficient to ensure  
1834 interoperability. It was constrained by the inability to use URI references to identify the SAML content to  
1835 be signed. With this limitation removed by the addition of SAML identifier attributes, a decision has been  
1836 made to forgo backwards compatibility with the older specification in this respect.

### 1837 **5.4.8 Example**

1838 Following is an example of a signed response containing a signed assertion. Line breaks have been  
1839 added for readability; the signatures are not valid and cannot be successfully verified.

```
1840 <Response  
1841   IssueInstant="2003-04-17T00:46:02Z"  
1842   MajorVersion="1"  
1843   MinorVersion="1"  
1844   Recipient="www.opensaml.org"
```

```

1845     ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"
1846     xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
1847     xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
1848     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1849     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
1850 <ds:Signature
1851   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1852 <ds:SignedInfo>
1853 <ds:CanonicalizationMethod
1854   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1855 <ds:SignatureMethod
1856   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1857 <ds:Reference
1858   URI="#_c7055387-af61-4fce-8b98-e2927324b306">
1859 <ds:Transforms>
1860 <ds:Transform
1861   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
1862 <ds:Transform
1863   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1864 <InclusiveNamespaces
1865   PrefixList="#default saml samlp ds xsd xsi"
1866   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1867 </ds:Transform>
1868 </ds:Transforms>
1869 <ds:DigestMethod
1870   Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1871 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
1872 </ds:Reference>
1873 </ds:SignedInfo>
1874 <ds:SignatureValue>
1875 x/GyPbzmFEe85pGD3claXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
1876 EIYcPzx+pX1h43SmwviCqXrjRtMANWbHLhWAptaKlywS7gFgsD01qjyen3CP+m3D
1877 w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>
1878 <ds:KeyInfo>
1879 <ds:X509Data>
1880 <ds:X509Certificate>
1881 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgaxCzAJBgNVBAYTA1VT
1882 MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
1883 F1VuaXZlcnNpdHkqb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
1884 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
1885 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVVoXDTA2MDkwNDA3Mjc1MVVowgYsX
1886 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
1887 Ym9yMQ4wDAYDVQQKEwVWVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuc2VzLmVz
1888 dTEhNCUGCSqGSIB3DQEEJARYYcm9vdEBzaGlMS5pbnRlcm5ldDIuZWR1MIGfMA0G
1889 CSqGSIB3DQEEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
1890 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIaOAPSZB113R6+KYiE7x4XAWIRCP+
1891 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
1892 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMASGA1UdDwQEAwIFoDANBgkq
1893 hkiG9w0BAQQFAAOBqQBfdqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
1894 qgi7lFV6MDkhhmTvTqBtjmnk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
1895 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
1896 </ds:X509Certificate>
1897 </ds:X509Data>
1898 </ds:KeyInfo>
1899 </ds:Signature>
1900 <Status><StatusCode Value="samlp:Success"/></Status>
1901 <Assertion
1902   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
1903   IssueInstant="2003-04-17T00:46:02Z"
1904   Issuer="www.opensaml.org"
1905   MajorVersion="1"
1906   MinorVersion="1"
1907   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
1908   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1909   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
1910 <Conditions
1911   NotBefore="2003-04-17T00:46:02Z"

```





---

## 1978 6 SAML Extensions

1979 The SAML schemas support extensibility. An example of an application that extends SAML assertions is  
1980 the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use  
1981 the extensibility features in SAML to create extension schemas.

1982 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML  
1983 elements can serve as the head element of a substitution group. However, SAML types are not defined  
1984 as *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only  
1985 elements and types that have been specifically designed to support extensibility.

### 1986 6.1 Assertion Schema Extension

1987 The SAML assertion schema is designed to permit separate processing of the assertion package and the  
1988 statements it contains, if the extension mechanism is used for either part.

1989 The following elements are intended specifically for use as extension points in an extension schema;  
1990 their types are set to *abstract*, and are thus usable only as the base of a derived type:

- 1991 • <Condition>
- 1992 • <Statement>
- 1993 • <SubjectStatement>

1994 The following elements that are directly usable as part of SAML MAY be extended:

- 1995 • <AuthenticationStatement>
- 1996 • <AuthorizationDecisionStatement>
- 1997 • <AttributeStatement>
- 1998 • <AudienceRestrictionCondition>

1999 The following elements are defined to allow elements from arbitrary namespaces within them, which  
2000 serves as a built-in extension point without requiring an extension schema:

- 2001 • <AttributeValue>
- 2002 • <Advice>

### 2003 6.2 Protocol Schema Extension

2004 The following SAML protocol elements are intended specifically for use as extension points in an  
2005 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived  
2006 type:

- 2007 • <Query>
- 2008 • <SubjectQuery>

2009 The following elements that are directly usable as part of SAML MAY be extended:

- 2010 • <Request>

- 2011 • <AuthenticationQuery>
- 2012 • <AuthorizationDecisionQuery>
- 2013 • <AttributeQuery>
- 2014 • <Response>

---

## 2015 7 SAML-Defined Identifiers

2016 The following sections define URI-based identifiers for common authentication methods, resource access  
2017 actions, and subject name identifier formats.

2018 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of  
2019 the most current RFC that specifies the protocol is used. URI references created specifically for SAML  
2020 have one of the following stems:

```
2021 urn:oasis:names:tc:SAML:1.0:  
2022 urn:oasis:names:tc:SAML:1.1:
```

### 2023 7.1 Authentication Method Identifiers

2024 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the  
2025 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although  
2026 both can refer to the same underlying mechanisms. An authentication statement with an  
2027 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The  
2028 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the  
2029 authentication statement does not provide the means to perform that authentication, such as a password,  
2030 key, or certificate.

2031 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,  
2032 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML  
2033 relying party to confirm that the request or message came from a system entity that corresponds to the  
2034 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method  
2035 that the relying party can use to do this in the future. This may or may not have any relationship to an  
2036 authentication that was performed previously. Unlike the authentication method, the subject confirmation  
2037 method may be accompanied by some piece of information, such as a certificate or key, that will allow  
2038 the relying party to perform the necessary check.

2039 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML  
2040 bindings and profiles specification [SAMLBind] for more information. Additional methods may be added  
2041 by defining new profiles or by private agreement.

2042 The following identifiers refer to SAML-specified authentication methods.

#### 2043 7.1.1 Password

2044 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

2045 The authentication was performed by means of a password.

#### 2046 7.1.2 Kerberos

2047 **URI:** urn:ietf:rfc:1510

2048 The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the  
2049 Needham-Schroeder symmetric key authentication mechanism [Needham78].

#### 2050 7.1.3 Secure Remote Password (SRP)

2051 **URI:** urn:ietf:rfc:2945

2052 The authentication was performed by means of Secure Remote Password protocol as specified in [RFC  
2053 2945].

#### 2054 **7.1.4 Hardware Token**

2055 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

2056 The authentication was performed using some (unspecified) hardware token.

#### 2057 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

2058 **URI:** urn:ietf:rfc:2246

2059 The authentication was performed using either the SSL or TLS protocol with certificate-based client  
2060 authentication. TLS is described in [RFC 2246].

#### 2061 **7.1.6 X.509 Public Key**

2062 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

2063 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2064 of an X.509 PKI [X.500][PKIX]. It may have been one of the mechanisms for which a more specific  
2065 identifier has been defined below.

#### 2066 **7.1.7 PGP Public Key**

2067 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

2068 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2069 of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more specific  
2070 identifier has been defined below.

#### 2071 **7.1.8 SPKI Public Key**

2072 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

2073 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2074 of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific identifier has  
2075 been defined below.

#### 2076 **7.1.9 XKMS Public Key**

2077 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

2078 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2079 of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more specific  
2080 identifier has been defined below.

#### 2081 **7.1.10 XML Digital Signature**

2082 **URI:** urn:ietf:rfc:3075

2083 The authentication was performed by means of an XML digital signature [RFC 3075].

2084 **7.1.11 Unspecified**

2085 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

2086 The authentication was performed by an unspecified means.

2087 **7.2 Action Namespace Identifiers**

2088 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element (see  
2089 Section Element `<Action>`) to refer to common sets of actions to perform on resources.

2090 **7.2.1 Read/Write/Execute/Delete/Control**

2091 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

2092 Defined actions:

2093 `Read Write Execute Delete Control`

2094 These actions are interpreted as follows:

2095 `Read`

2096 The subject may read the resource.

2097 `Write`

2098 The subject may modify the resource.

2099 `Execute`

2100 The subject may execute the resource.

2101 `Delete`

2102 The subject may delete the resource.

2103 `Control`

2104 The subject may specify the access control policy for the resource.

2105 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

2106 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

2107 Defined actions:

2108 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

2109 The actions specified in Section `Read/Write/Execute/Delete/Control` are interpreted in the same manner  
2110 described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively  
2111 specify that the stated permission is denied. Thus a subject described as being authorized to perform the  
2112 action `~Read` is affirmatively denied read permission.

2113 A SAML authority MUST NOT authorize both an action and its negated form.

2114 **7.2.3 Get/Head/Put/Post**

2115 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

2116 Defined actions:

2117 GET HEAD PUT POST

2118 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
2119 the GET action on a resource is authorized to retrieve it.

2120 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and  
2121 POST actions to the write permission. The correspondence is not exact however since an HTTP GET  
2122 operation may cause data to be modified and a POST operation may cause modification to a resource  
2123 other than the one specified in the request. For this reason a separate Action URI reference specifier is  
2124 provided.

## 2125 7.2.4 UNIX File Permissions

2126 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

2127 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
2128 notation.

2129 The action string is a four-digit numeric code:

2130 *extended user group world*

2131 Where the *extended* access permission has the value

2132 +2 if sgid is set

2133 +4 if suid is set

2134 The *user group* and *world* access permissions have the value

2135 +1 if execute permission is granted

2136 +2 if write permission is granted

2137 +4 if read permission is granted

2138 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read  
2139 and execute; and world read.

## 2140 7.3 NameIdentifier Format Identifiers

2141 The following identifiers MAY be used in the `Format` attribute of the `<NameIdentifier>` element (see  
2142 Section ) to refer to common formats for the content of the `<NameIdentifier>` element and the  
2143 associated processing rules, if any.

2144 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of  
2145 SAML.

### 2146 7.3.1 Unspecified

2147 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

2148 The interpretation of the content of the element is left to individual implementations.

## 2149 **7.3.2 Email Address**

2150 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

2151 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as  
2152 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that  
2153 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in  
2154 parentheses) after it, and is not surrounded by "<" and ">".

## 2155 **7.3.3 X.509 Subject Name**

2156 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

2157 Indicates that the content of the element is in the form specified for the contents of the  
2158 <ds:X509SubjectName> element in the XML Signature Recommendation . Implementors should note  
2159 that the XML Signature specification specifies encoding rules for X.509 subject names that differ from the  
2160 rules given in IETF RFC 2253 [RFC 2253].

## 2161 **7.3.4 Windows Domain Qualified Name**

2162 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

2163 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
2164 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"  
2165 separator MAY be omitted.

## 2166 **7.3.5 Provider Identifier**

2167 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:provider

2168 Indicates that the content of the element is the identifier of a provider of SAML-based services (such as a  
2169 SAML authority) or a participant in SAML profiles (such as a service provider supporting the browser  
2170 profiles). Such an identifier can be used to make assertions about system entities that can issue SAML  
2171 requests, responses, and assertions.

## 2172 **7.3.6 Federated Identifier**

2173 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:federated

2174 Indicates that the content of the element is a persistent opaque identifier that corresponds to an identity  
2175 federation between an identity provider and a service provider (or affiliation of service providers).  
2176 Federated name identifiers generated by identity providers MUST be constructed using pseudo-random  
2177 values that have no discernible correspondence with the subject's actual identifier (for example,  
2178 username). The intent is to create a non-public pseudonym to prevent the discovery of the subject's  
2179 identity or activities. Federated name identifier values MUST NOT exceed a length of 256 characters.

2180 The element's content MUST contain the most recent identifier of the subject set by the identity provider.

2181 The element's `NameQualifier` attribute, if present, MUST contain the name of the identity provider  
2182 participating in the identity federation. It MAY be omitted if the value can be derived from the context of  
2183 the message containing the element, such as the issuer of an assertion.

2184 The element's `SPNameQualifier` attribute, if present, MUST contain the name of the service provider  
2185 or affiliation of providers participating in the identity federation. It MAY be omitted if the element is

2186 contained in a message intended only for consumption directly by the service provider, and the value  
2187 would be the name of that service provider.

2188 The element's `SPProvidedIdentifier` attribute MUST contain the alternative identifier of the subject  
2189 most recently set by the service provider or affiliation, if any. If no such identifier has been established,  
2190 than the attribute MUST be omitted.

2191 Federated identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear  
2192 text with providers other than the providers that have established the identity federation. Furthermore,  
2193 they MUST NOT appear in log files or similar locations without appropriate controls and protections.  
2194 Deployments without such requirements are free to use other kinds of identifiers in their SAML  
2195 exchanges.

### 2196 **7.3.7 Transient Identifier**

2197 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

2198 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated  
2199 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated  
2200 in accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of  
2201 256 characters.

2202 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier  
2203 represents a transient and temporary identity federation, as described in Section Federated Identifier. In  
2204 such a case, they MAY be omitted in accordance with the rules specified in that section.



2205

## 8 References

2206 The following works are cited in the body of this specification.

### 8.1 Normative References

- 2208       **[Excl-C14N]**       J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web  
2209                       Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 2210       **[Schema1]**       H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
2211                       Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.  
2212                       Note that this specification normatively references [Schema2], listed below.
- 2213       **[Schema2]**       P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
2214                       Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 2215       **[XML]**            T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World  
2216                       Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 2217       **[XMLEnc]**        D. Eastlake et al., XML Encryption Syntax and Processing,  
2218                       <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web  
2219                       Consortium. Note that this specification normatively references , listed below.
- 2220       **[XMLEnc-XSD]**   XML Encryption Schema. World Wide Web Consortium.  
2221                       <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 2222       **[XMLNS]**        T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January  
2223                       1999. <http://www.w3.org/TR/REC-xml-names>.
- 2224       **[XMLSig]**        D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web  
2225                       Consortium, February 2002. <http://www.w3.org/TR/xmldsig-core/>. Note that this  
2226                       specification normatively references [XMLSig-XSD], listed below.
- 2227       **[XMLSig-XSD]**   XML Signature Schema. World Wide Web Consortium.  
2228                       [http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-](http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd)  
2229                       [schema.xsd](http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd).

### 8.2 Non-Normative References

- 2231       **[LibertyProt]**   J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty  
2232                       Alliance Project, January 2003,  
2233                       [http://www.projectliberty.org/specs/archive/v1\\_1/liberty-architecture-protocols-](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf)  
2234                       [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 2235       **[Needham78]**    R. Needham et al. *Using Encryption for Authentication in Large Networks of*  
2236                       *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December  
2237                       1978.
- 2238       **[PGP]**            Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*.  
2239                       IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 2240       **[PKIX]**            R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*  
2241                       *Certificate and CRL Profile*. IETF RFC 2459, January 1999.  
2242                       <http://www.ietf.org/rfc/rfc2459.txt>.
- 2243       **[RFC 1510]**        J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.  
2244                       IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 2245       **[RFC 2119]**        S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF  
2246                       RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

2247 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January  
2248 1999. <http://www.ietf.org/rfc/rfc2246.txt>.

2249 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*  
2250 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.  
2251 <http://www.ietf.org/rfc/rfc2253.txt>.

2252 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF  
2253 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

2254 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.  
2255 <http://www.ietf.org/rfc/rfc2630.txt>.

2256 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.  
2257 <http://www.ietf.org/rfc/rfc2822.txt>.

2258 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,  
2259 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

2260 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF  
2261 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

2262 [SAMLBind] E. Maler et al. *Bindings and Profiles for the OASIS Security Assertion Markup*  
2263 *Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-  
2264 bindings-1.1. <http://www.oasis-open.org/committees/security/>.

2265 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*  
2266 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
2267 oasis-sstc-saml-conform-1.1. HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>2268 open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

2269 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*  
2270 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-  
open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-<br/>2271 open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

2272 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*  
2273 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1.  
2274 HYPERLINK "[http://www.oasis-  
2275 open.org/committees/security/](http://www.oasis-open.org/committees/security/)"[http://www.oasis-  
open.org/committees/security/](http://www.oasis-open.org/committees/security/).

2276 [SAMPL-XSD] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID  
2277 oasis-sstc-saml-schema-protocol-1.1. HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>2278 open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

2279 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*  
2280 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
2281 oasis-sstc-saml-sec-consider-1.1. HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>2282 open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

2283 [SAML-XSD] E. Maler et al. *SAML assertion schema*. OASIS, September 2003. Document ID  
2284 oasis-sstc-saml-schema-assertion-1.1. HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>2285 open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

2286 [SPKI] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. *SPKI*  
2287 *Certificate Theory*. IETF RFC 2693, September 1999.  
2288 <http://www.ietf.org/rfc/rfc2693.txt>.

2289 [UNICODE-C] M. Davis, M. J. Dürst. *Unicode Normalization Forms*. UNICODE Consortium,  
2290 March 2001. <http://www.unicode.org/unicode/reports/tr15/tr15-21.html>.

2291 [W3C-CHAR] M. J. Dürst. *Requirements for String Identity Matching and String Indexing*. World  
2292 Wide Web Consortium, July 1998. <http://www.w3.org/TR/WD-charreq>.

2293 [W3C-CharMod] M. J. Dürst. *Character Model for the World Wide Web 1.0*. World Wide Web  
2294 Consortium, April, 2002. <http://www.w3.org/TR/charmod/>.

2295       **[X.500]**           ITU-T Recommendation X.501: Information Technology - Open Systems  
2296                           Interconnection - The Directory: Models. 1993.

2297       **[XKMS]**           W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J.  
2298                           Lapp. XML Key Management Specification (XKMS). W3C Note 30 March 2001.  
2299                           <http://www.w3.org/TR/xkms/>.

---

2300 **Appendix A. Acknowledgments**

2301 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
2302 Committee, whose voting members at the time of publication were:

- 2303 • @@

## Appendix B. Revision History

Rev	Date	By Whom	What
01	20 Oct 2003	Eve Maler	Initial draft. Converted to OpenOffice. <b>CORE-1</b> through <b>CORE-4</b> . Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed.
02	4 Jan 2004	Eve Maler	Implemented Scott Cantor's draft-sstc-nameid-07 solution proposal ( <a href="http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587">http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587</a> ) for work item <b>W-2</b> , Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet).  Fixed <b>CORE-10</b> (the description of subelement occurrence in the <Evidence> element).
03	24 Jan 2004	Scott Cantor	Name identifier, issuer, and federation protocol additions/changes. See 03-interim-diff draft for intermediate set of change bars.
04	1 Feb 2004	Eve Maler	Made minor edits to new and existing material; changed new <AssertionRequest> element name to <AssertionIDRequest>; changed new <AssertionArtifact> and <NewIdentifier> element declarations from local to global; made distinction between normative and non-normative references; implemented the blocking of element substitution. The bulk of work item W-2, Identity Federation, is now reflected here. What remains is the federation termination protocol, plus a few other pieces that are covered under other work items.
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol, removed NameID date attributes, clarified Name Reg processing rules, added Extensions facility and Consent attribute. Also moved Signature on assertions to a location consistent with Request and Response. Added session protocol material; still unfinished.

2306

---

## Appendix C. Notices

2307 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
2308 might be claimed to pertain to the implementation or use of the technology described in this document or  
2309 the extent to which any license under such rights might or might not be available; neither does it  
2310 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
2311 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
2312 made available for publication and any assurances of licenses to be made available, or the result of an  
2313 attempt made to obtain a general license or permission for the use of such proprietary rights by  
2314 implementors or users of this specification, can be obtained from the OASIS Executive Director.

2315 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
2316 or other proprietary rights which may cover technology that may be required to implement this  
2317 specification. Please address the information to the OASIS Executive Director.

2318 **Copyright © OASIS Open 2004. All Rights Reserved.**

2319 This document and translations of it may be copied and furnished to others, and derivative works that  
2320 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
2321 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright  
2322 notice and this paragraph are included on all such copies and derivative works. However, this document  
2323 itself may not be modified in any way, such as by removing the copyright notice or references to OASIS,  
2324 except as needed for the purpose of developing OASIS specifications, in which case the procedures for  
2325 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required  
2326 to translate it into languages other than English.

2327 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
2328 or assigns.

2329 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2330 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
2331 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS  
2332 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR  
2333 PURPOSE.