



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

Working Draft **098**, ~~15-March~~**8 April** 2004

**Document identifier:**

sstc-saml-core-2.0-draft-**098**

**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  
**John Hughes, Entegri**  
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)

38 **Abstract:**

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

41 **Status:**

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

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

52 For information on whether any patents have been disclosed that may be essential to  
53 implementing this specification, and any offers of patent licensing terms, please refer to the  
54 Intellectual Property Rights web page for the Security Services TC ([http://www.oasis-  
open.org/committees/security/ipr.php](http://www.oasis-<br/>55 open.org/committees/security/ipr.php)).

# 56 Table of Contents

57	1 Introduction.....	7
58	1.1 Notation.....	7
59	1.2 Schema Organization and Namespaces.....	8
60	1.2.1 String and URI Values.....	8
61	1.2.2 Time Values.....	8
62	1.2.3 ID and ID Reference Values.....	8
63	1.2.4 Comparing SAML Values.....	9
64	2 SAML Assertions.....	10
65	2.1 Schema Header and Namespace Declarations.....	10
66	2.2 Simple Types.....	10
67	2.2.1 Simple Type DecisionType.....	11
68	2.3 Name Identifiers.....	11
69	2.3.1 Element <BaseIdentifier>.....	11
70	2.3.2 Element <NameIdentifier>.....	12
71	2.3.3 Element <EncryptedIdentifier>.....	12
72	2.3.4 Element <Issuer>.....	13
73	2.4 Assertions.....	13
74	2.4.1 Element <AssertionIDReference>.....	13
75	2.4.2 Element <AssertionURIReference>.....	13
76	2.4.3 Element <Assertion>.....	14
77	2.4.3.1 Element <Subject>.....	15
78	2.4.3.2 Element <Conditions>.....	16
79	2.4.3.2.1 Attributes NotBefore and NotOnOrAfter.....	17
80	2.4.3.2.2 Element <Condition>.....	17
81	2.4.3.2.3 Elements <AudienceRestrictionCondition> and <Audience>.....	17
82	2.4.3.2.4 Element <DoNotCacheCondition>.....	18
83	2.4.3.2.5 Element <ProxyRestrictionCondition>.....	19
84	2.4.3.3 Element <Advice>.....	20
85	2.5 Statements.....	20
86	2.5.1 Element <Statement>.....	20
87	2.5.1.1 Elements <SubjectConfirmation>, <ConfirmationMethod>, and	
88	<SubjectConfirmationData>.....	21
89	2.5.2 Element <AuthenticationStatement>.....	21
90	2.5.2.1 Element <SubjectLocality>.....	22
91	2.5.2.2 Element <AuthnContext>.....	23
92	2.5.3 Element <AttributeStatement>.....	23
93	2.5.3.1 Elements <AttributeDesignator> and <Attribute>.....	23
94	2.5.3.1.1 Element <AttributeValue>.....	25
95	2.5.4 Element <AuthorizationDecisionStatement>.....	25
96	2.5.4.1 Element <Action>.....	27
97	2.5.4.2 Element <Evidence>.....	27
98	3 SAML Protocols.....	29
99	3.1 Schema Header and Namespace Declarations.....	29
100	3.2 Requests and Responses.....	30
101	3.2.1 Complex Type RequestAbstractType.....	30
102	3.2.1.1 Element <RelayState>.....	31
103	3.2.2 Complex Type StatusResponseType.....	31
104	3.2.2.1 Element <Status>.....	32

105	3.2.2.2 Element <StatusCode>.....	33
106	3.2.2.3 Element <StatusMessage>.....	36
107	3.2.2.4 Element <StatusDetail>.....	36
108	3.3 Assertion Query and Request Protocol.....	36
109	3.3.1 Element <AssertionIDRequest>.....	36
110	3.3.2 Queries.....	36
111	3.3.2.1 Element <SubjectQuery>.....	37
112	3.3.2.2 Element <AuthenticationQuery>.....	37
113	3.3.2.3 Element <AttributeQuery>.....	38
114	3.3.2.4 Element <AuthorizationDecisionQuery>.....	39
115	3.3.3 Element <Response>.....	40
116	3.3.3.1 Processing Rules.....	40
117	3.4 Authentication Request Protocol.....	40
118	3.4.1 Element <AuthnRequest>.....	41
119	3.4.1.1 Element <NameIDPolicy>.....	43
120	3.4.1.2 Element <RequestAuthnContext>.....	44
121	3.4.1.3 Element <Scoping>.....	45
122	3.4.1.4 Element <IDPList>.....	45
123	3.4.1.5 Element <IDPEntity>.....	46
124	3.4.1.6 Processing Rules.....	46
125	3.4.1.7 Proxying.....	47
126	3.4.1.7.1 Processing Rules.....	47
127	3.5 Artifact Protocol.....	49
128	3.5.1 Element <ArtifactRequest>.....	49
129	3.5.2 Element <ArtifactResponse>.....	50
130	3.5.3 Processing Rules.....	50
131	3.6 Federated Name Registration Protocol.....	51
132	3.6.1 Element <RegisterNameIdentifierRequest>.....	51
133	3.6.2 Element <RegisterNameIdentifierResponse>.....	52
134	3.6.3 Processing Rules.....	52
135	3.7 Federation Termination Protocol.....	53
136	3.7.1 Element <FederationTerminationNotification>.....	53
137	3.7.2 Element <FederationTerminationResponse>.....	54
138	3.7.3 Processing Rules.....	54
139	3.8 Single Logout Protocol.....	54
140	3.8.1 Element <LogoutRequest>.....	55
141	3.8.2 Element <LogoutResponse>.....	55
142	3.8.3 Processing Rules.....	56
143	3.8.3.1 Session Participant Rules.....	56
144	3.8.3.2 Session Authority Rules.....	56
145	3.9 Name Identifier Mapping Protocol.....	57
146	3.9.1 Element <NameIdentifierMappingRequest>.....	57
147	3.9.2 Element <NameIdentifierMappingResponse>.....	58
148	3.9.3 Processing Rules.....	59
149	4 SAML Versioning.....	60
150	4.1 SAML Specification Set Version.....	60
151	4.1.1 Schema Version.....	60
152	4.1.2 SAML Assertion Version.....	60
153	4.1.3 SAML Protocol Version.....	61
154	4.1.3.1 Request Version.....	61
155	4.1.4 Response Version.....	61
156	4.1.5 Permissible Version Combinations.....	62

157	4.2 SAML Namespace Version.....	62
158	4.2.1 Schema Evolution.....	62
159	5 SAML and XML Signature Syntax and Processing.....	63
160	5.1 Signing Assertions.....	64
161	5.2 Request/Response Signing.....	64
162	5.3 Signature Inheritance.....	64
163	5.4 XML Signature Profile.....	64
164	5.4.1 Signing Formats and Algorithms.....	64
165	5.4.2 References.....	64
166	5.4.3 Canonicalization Method.....	65
167	5.4.4 Transforms.....	65
168	5.4.5 KeyInfo.....	65
169	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	65
170	5.4.8 Example.....	65
171	6 SAML Extensions.....	68
172	6.1 Assertion Schema Extension.....	68
173	6.2 Protocol Schema Extension.....	68
174	7 SAML-Defined Identifiers.....	70
175	7.1 Authentication Method Identifiers.....	70
176	7.1.1 Password.....	70
177	7.1.2 Kerberos .....	70
178	7.1.3 Secure Remote Password (SRP).....	70
179	7.1.4 Hardware Token.....	71
180	7.1.5 SSL/TLS Certificate Based Client Authentication.....	71
181	7.1.6 X.509 Public Key .....	71
182	7.1.7 PGP Public Key .....	71
183	7.1.8 SPKI Public Key .....	71
184	7.1.9 XKMS Public Key .....	71
185	7.1.10 XML Digital Signature .....	71
186	7.1.11 Authentication Context.....	72
187	7.1.12 Unspecified .....	72
188	7.2 Action Namespace Identifiers.....	72
189	7.2.1 Read/Write/Execute/Delete/Control.....	72
190	7.2.2 Read/Write/Execute/Delete/Control with Negation.....	72
191	7.2.3 Get/Head/Put/Post.....	73
192	7.2.4 UNIX File Permissions.....	73
193	7.3 NameIdentifier Format Identifiers.....	73
194	7.3.1 Unspecified.....	74
195	7.3.2 Email Address.....	74
196	7.3.3 X.509 Subject Name.....	74
197	7.3.4 Windows Domain Qualified Name.....	74
198	7.3.5 Provider Identifier.....	74
199	7.3.6 Federated Identifier.....	74
200	7.3.7 Transient Identifier.....	75
201	7.4 Attribute NameFormat Identifiers.....	75
202	7.4.1 Unspecified.....	75
203	7.4.2 URI Reference.....	75
204	7.5 Attribute ValueType Identifiers.....	76
205	7.5.1 Application-SpecificUnspecified Value Type.....	76
206	8 References.....	77

207	8.1 Normative References.....	77
208	8.2 Non-Normative References.....	77
209		

---

# 1 Introduction

210

211 This specification defines the syntax and semantics for Security Assertion Markup Language (SAML)  
212 assertions and the protocols for requesting and returning them. SAML assertions, requests, and  
213 responses are encoded in XML [XML]and use XML namespaces [XMLNS]. They are typically embedded  
214 in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The  
215 SAML specification for bindings [SAMLBind] provides frameworks for this embedding and transport. Files  
216 containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAML-P-XSD] are  
217 available.

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

## 1.1 Notation

219

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

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

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

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

230         Listings of SAML schemas appear like this.

231         Example code listings appear like this.

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

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

- 238         • The prefix `saml:` stands for the SAML assertion namespace,  
239         `urn:oasis:names:tc:SAML:2.0:assertion`.
- 240         • The prefix `samlp:` stands for the SAML request-response protocol namespace,  
241         `urn:oasis:names:tc:SAML:2.0:protocol`.
- 242         • The prefix `ds:` stands for the W3C XML Signature namespace,  
243         <http://www.w3.org/2000/09/xmldsig#> [XMLSig-XSD].
- 244         • The prefix `xenc:` stands for the W3C XML Encryption namespace,  
245         <http://www.w3.org/2001/04/xmlenc#> [XMLEnc-XSD].
- 246         • The prefix `xsd:` stands for the W3C XML Schema namespace,  
247         <http://www.w3.org/2001/XMLSchema> [Schema1], in example listings. In schema listings, this is  
248         the default namespace and no prefix is shown.

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

## 251 1.2 Schema Organization and Namespaces

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

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

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

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

258 The assertion schema is imported into the protocol schema. Also imported into both schemas is the  
259 schema for XML Signature [XMLSig], which is associated with the following XML namespace:

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

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

### 262 1.2.1 String and URI Values

263 All SAML string and URI reference values have the types **xsd:string** and **xsd:anyURI** respectively,  
264 which are built in to the W3C XML Schema Datatypes specification [Schema2]. All strings in SAML  
265 messages MUST consist of at least one non-whitespace character (whitespace is defined in the XML  
266 Recommendation [XML]§2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise  
267 indicated in this specification, all URI reference values MUST consist of at least one non-whitespace  
268 character, and are REQUIRED to be absolute [RFC 2396].

### 269 1.2.2 Time Values

270 All SAML time values have the type **xsd:dateTime**, which is built in to the W3C XML Schema Datatypes  
271 specification [Schema1], and MUST be expressed in UTC form.

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

### 274 1.2.3 ID and ID Reference Values

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

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

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

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

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

## 292 **1.2.4 Comparing SAML Values**

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

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

304 Applications that compare data received in SAML documents to data from external sources **MUST** take  
305 into account the normalization rules specified for XML. Text contained within elements is normalized so  
306 that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the  
307 XML Recommendation [XML]§2.11. Attribute values defined as strings (or types derived from strings) are  
308 normalized as described in [XML] §3.3.3. All white space characters are replaced with blanks (ASCII code  
309 32<sub>Decimal</sub>).

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

---

## 2 SAML Assertions

313

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

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

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

### 2.1 Schema Header and Namespace Declarations

327

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

```
330 <schema
331     targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
332     xmlns="http://www.w3.org/2001/XMLSchema"
333     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
334     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
335     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
336     elementFormDefault="unqualified"
337     attributeFormDefault="unqualified"
338     blockDefault="substitution"
339     version="2.0">
340     <import namespace="http://www.w3.org/2000/09/xmldsig#"
341           schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
342 schema.xsd"/>
343     <import namespace="http://www.w3.org/2001/04/xmlenc#"
344           schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
345 schema.xsd"/>
346     <annotation>
347         <documentation>
348             Document identifier: sstc-saml-schema-assertion-2.0
349             Location: http://www.oasis-
350 open.org/committees/documents.php?wg_abbrev=security
351         </documentation>
352     </annotation>
353     ...
354 </schema>
```

### 2.2 Simple Types

355

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

## 357 2.2.1 Simple Type DecisionType

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

360 Permit

361 The specified action is permitted.

362 Deny

363 The specified action is denied.

364 Indeterminate

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

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

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

```
370 <simpleType name="DecisionType">  
371   <restriction base="string">  
372     <enumeration value="Permit"/>  
373     <enumeration value="Deny"/>  
374     <enumeration value="Indeterminate"/>  
375   </restriction>  
376 </simpleType>
```

## 377 2.3 Name Identifiers

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

### 380 2.3.1 Element <BaseIdentifier>

381 The `<BaseIdentifier>` element is an extension point that allows applications to add new kinds of  
382 identifiers. Its **BaseIdentifierAbstractType** complex type is abstract and is thus usable only as the base  
383 of a derived type. It defines the following common attributes for all identifier representations:

384 NameQualifier [Optional]

385 The security or administrative domain that qualifies the identifier of the subject. This attribute  
386 provides a means to federate identifiers from disparate user stores without collision.

387 SPNameQualifier [Optional]

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

390 The following schema fragment defines the `<BaseIdentifier>` element and its **BaseIdentifierType**  
391 complex type:

```
392 <element name="BaseIdentifier" type="saml:BaseIdentifierAbstractType"/>  
393 <complexType name="BaseIdentifierAbstractType" abstract="true">  
394   <complexContent>  
395     <extension base="anyType">  
396       <attribute name="NameQualifier" type="string" use="optional"/>  
397       <attribute name="SPNameQualifier" type="string" use="optional"/>  
398     </extension>
```

```
399     </complexContent>
400 </complexType>
```

### 401 2.3.2 Element <NameIdentifier>

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

404 Format [Optional]

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

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

415 SPProvidedIdentifier [Optional]

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

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

```
421 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
422 <complexType name="NameIdentifierType" mixed="false">
423   <simpleContent>
424     <restriction base="saml:BaseIdentifierAbstractType">
425       <simpleType>
426         <restriction base="string"/>
427       </simpleType>
428       <attribute name="Format" type="anyURI" use="optional"/>
429       <attribute name="SPProvidedIdentifier" type="string"
430 use="optional"/>
431     </restriction>
432   </simpleContent>
433 </complexType>
```

### 434 2.3.3 Element <EncryptedIdentifier>

435 The <EncryptedIdentifier> element extends **BaseIdentifierAbstractType** to carry the content of  
436 the element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification  
437 [XMLEnc]. The <EncryptedIdentifier> element contains the following additional elements and  
438 attributes:

439 <xenc:EncryptedData> [Required]

440 The encrypted content and associated encryption details, as defined by . The encrypted content  
441 MUST contain an element that has a type that is derived from **BaseIdentifierAbstractType** or  
442 from **AssertionType**.

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

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

449 The following schema fragment defines the <EncryptedIdentifier> element and its  
450 **EncryptedIdentifierType** complex type:

```
451     <element name="EncryptedIdentifier" type="saml:EncryptedIdentifierType"/>  
452     <complexType name="EncryptedIdentifierType" mixed="false">  
453       <complexContent>  
454         <restriction base="saml:BaseIdentifierType">  
455           <sequence>  
456             <element ref="xenc:EncryptedData"/>  
457             <element ref="xenc:EncryptedKey" minOccurs="0"  
458 maxOccurs="unbounded"/>  
459           </sequence>  
460         </restriction>  
461       </complexContent>  
462     </complexType>
```

### 463 2.3.4 Element <Issuer>

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

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

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

## 469 2.4 Assertions

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

### 471 2.4.1 Element <AssertionIDReference>

472 The <AssertionIDReference> element makes a reference to a SAML assertion by its unique  
473 identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is  
474 not specified as part of the reference.

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

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

### 477 2.4.2 Element <AssertionURIReference>

478 The <AssertionURIReference> element makes a reference to a SAML assertion by its uniform  
479 resource identifier (URI). Dereferencing the URI (in a fashion dictated by the URI) is intended to produce  
480 the assertion. [See the Bindings specification \[SAMLBind\] for information on how this element is used in a  
481 protocol binding.](#)

482 The following schema fragment defines the <AssertionURIReference> element:

483 `<element name="AssertionURIReference" type="anyURI"/>`

### 484 2.4.3 Element <Assertion>

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

487 **MajorVersion** [Required]

488 The major version of this assertion. The identifier for the version of SAML defined in this  
489 specification is 2. SAML versioning is discussed in Section SAML Versioning.

490 **MinorVersion** [Required]

491 The minor version of this assertion. The identifier for the version of SAML defined in this  
492 specification is 0. SAML versioning is discussed in Section SAML Versioning.

493 **AssertionID** [Required]

494 The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in  
495 Section 1.2.3 for identifier uniqueness.

496 **IssueInstant** [Required]

497 The time instant of issue in UTC, as described in Section Time Values.

498 **<Issuer>** [Required]

499 The SAML authority that is making the claim(s) in the assertion. The issuer identity SHOULD be  
500 unambiguous to the intended relying parties. If the Format attribute is omitted, the identifier  
501 `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see section 7.3.1) is  
502 assumed.

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

506 **<ds:Signature>** [Optional]

507 [An XML Signature that authenticates the assertion, as described in Section SAML and XML](#)  
508 [Signature Syntax and Processing.](#)

509 **<Subject>** [Required]

510 The subject of the statement(s) in the assertion.

511 ~~**<ds:Signature>** [Optional]~~

512 ~~[An XML Signature that authenticates the assertion, as described in Section SAML and XML](#)~~  
513 ~~[Signature Syntax and Processing.](#)~~

514 **<Conditions>** [Optional]

515 Conditions that MUST be taken into account in assessing the validity of and/or using the assertion.

516 **<Advice>** [Optional]

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

519 One or more of the following statement elements:

520 **<Statement>**

521 A statement defined in an extension schema.

522 <AuthenticationStatement>  
 523     An authentication statement.  
 524 <AuthorizationDecisionStatement>  
 525     An authorization decision statement.  
 526 <AttributeStatement>  
 527     An attribute statement.

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

```

530 <element name="Assertion" type="saml:AssertionType"/>
531 <complexType name="AssertionType">
532   <sequence>
533     <element ref="saml:Issuer"/>
534     <element ref="ds:Signature" minOccurs="0"/>
535     <element ref="saml:Subject"/>
536     <element ref="ds:Signature" minOccurs="0"/>
537     <element ref="saml:Conditions" minOccurs="0"/>
538     <element ref="saml:Advice" minOccurs="0"/>
539     <choice maxOccurs="unbounded">
540       <element ref="saml:Statement"/>
541       <element ref="saml:AuthenticationStatement"/>
542       <element ref="saml:AuthorizationDecisionStatement"/>
543       <element ref="saml:AttributeStatement"/>
544     </choice>
545   </sequence>
546   <attribute name="MajorVersion" type="integer" use="required"/>
547   <attribute name="MinorVersion" type="integer" use="required"/>
548   <attribute name="AssertionID" type="ID" use="required"/>
549   <attribute name="IssueInstant" type="dateTime" use="required"/>
550 </complexType>

```

### 551 2.4.3.1 Element <Subject>

552 The <Subject> element specifies the principal that is the subject of all of the (one or more) statements  
 553 in the assertion. It contains a name identifier, a series of one or more subject confirmations, or both:

554 <NameIdentifier>, <EncryptedIdentifier>, or <BaseIdentifier>  
 555     Identifies the subject.†

556 <SubjectConfirmation>  
 557     Information that allows the subject to be authenticated. If more than one subject confirmation is  
 558     provided, then usage of any one of them is sufficient to confirm the subject for the purpose of  
 559     applying the assertion.

560 If the <Subject> element contains both an identifier and one or more subject confirmations, the SAML  
 561 authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it  
 562 can treat the entity presenting the assertion to the relying party as the entity that the SAML authority  
 563 associates with the name identifier. A <Subject> element SHOULD NOT identify more than one  
 564 principal.

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

```

566 <element name="Subject" type="saml:SubjectType"/>
567 <complexType name="SubjectType">
568   <choice>
569     <sequence>
570       <choice>

```

```

571         <element ref="saml:BaseIdentifier"/>
572         <element ref="saml:NameIdentifier"/>
573         <element ref="saml:EncryptedIdentifier"/>
574     </choice>
575     <element ref="saml:SubjectConfirmation" minOccurs="0"
576 maxOccurs="unbounded"/>
577 </sequence>
578 <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
579 </choice>
580 </complexType>

```

### 581 2.4.3.2 Element <Conditions>

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

583 NotBefore [Optional]

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

586 NotOnOrAfter [Optional]

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

589 <Condition> [Any Number]

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

591 <AudienceRestrictionCondition> [Any Number]

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

593 <DoNotCacheCondition> [Any Number]

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

596 <ProxyRestrictionCondition> [Any Number]

597 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent  
598 assertions of their own on the basis of the information contained in the original assertion.

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

```

601 <element name="Conditions" type="saml:ConditionsType"/>
602 <complexType name="ConditionsType">
603     <choice minOccurs="0" maxOccurs="unbounded">
604         <element ref="saml:AudienceRestrictionCondition"/>
605         <element ref="saml:DoNotCacheCondition">
606         <element ref="saml:ProxyRestrictionCondition"/>
607         <element ref="saml:Condition"/>
608     </choice>
609     <attribute name="NotBefore" type="dateTime" use="optional"/>
610     <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
611 </complexType>

```

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

- 616 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is  
617 considered to be **Valid**.

- 618 2. If any sub-element or attribute of the `<Conditions>` element is determined to be invalid, then the  
619 assertion is **Invalid**.
- 620 3. If any sub-element or attribute of the `<Conditions>` element cannot be evaluated, then the validity  
621 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 622 4. If all sub-elements and attributes of the `<Conditions>` element are determined to be **Valid**, then  
623 the assertion is considered to be **Valid**.
- 624 The `<Conditions>` element MAY be extended to contain additional conditions. If an element contained  
625 within a `<Conditions>` element is encountered that is not understood, the status of the condition  
626 cannot be evaluated and the validity status of the assertion MUST be deemed to be **Indeterminate** in  
627 accordance with rule 3 above.
- 628 Note that an assertion that has validity status **Valid** may not be trustworthy for reasons such as not being  
629 issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.
- 630 Also note that some conditions may not directly impact the validity of the containing assertion (they  
631 always evaluate to **Valid**), but may restrict the behavior of relying parties with respect to the use of the  
632 assertion.

#### 633 **2.4.3.2.1 Attributes NotBefore and NotOnOrAfter**

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

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

637 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the  
638 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the  
639 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the  
640 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),  
641 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither  
642 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is  
643 valid at any time.

644 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is  
645 built in to the W3C XML Schema Datatypes specification [Schema2]. All time instants are specified in  
646 Universal Coordinated Time (UTC) as described in Section Time Values.

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

#### 648 **2.4.3.2.2 Element <Condition>**

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

651 The following schema fragment defines the `<Condition>` element and its **ConditionAbstractType**  
652 complex type:

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

#### 655 **2.4.3.2.3 Elements <AudienceRestrictionCondition> and <Audience>**

656 The `<AudienceRestrictionCondition>` element specifies that the assertion is addressed to one or  
657 more specific audiences identified by `<Audience>` elements. Although a SAML relying party that is

658 outside the audiences specified is capable of drawing conclusions from an assertion, the SAML authority  
659 explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the  
660 following elements:

661 <Audience>

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

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

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

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

```
674 <element name="AudienceRestrictionCondition"  
675 type="saml:AudienceRestrictionConditionType"/>  
676 <complexType name="AudienceRestrictionConditionType">  
677 <complexContent>  
678 <extension base="saml:ConditionAbstractType">  
679 <sequence>  
680 <element ref="saml:Audience" maxOccurs="unbounded"/>  
681 </sequence>  
682 </extension>  
683 </complexContent>  
684 </complexType>  
685 <element name="Audience" type="anyURI"/>
```

#### 686 2.4.3.2.4 Element <DoNotCacheCondition>

687 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be  
688 retained for future use. Note that no relying party is required to perform caching. However, any that do so  
689 MUST observe this condition. This condition conveys one-time-use semantics, and is independent from  
690 the NotBefore and NotOnOrAfter condition information.

691 A SAML authority SHOULD NOT include more than one <DoNotCacheCondition> element within a  
692 <Conditions> element of an assertion. If multiple <DoNotCacheCondition> elements appear within  
693 a <Conditions> element, a Relying Party MUST treat the multiple elements as though a single  
694 <DoNotCacheCondition> element was specified.

695 For the purposes of determining the validity of the <Conditions> element, the  
696 <DoNotCacheCondition> is considered to always be valid.

697 The following schema fragment defines the <DoNotCacheCondition> element and its  
698 **DoNotCacheConditionType** complex type:

```
699 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType"/>  
700 <complexType name="DoNotCacheConditionType">  
701 <complexContent>  
702 <extension base="saml:ConditionAbstractType"/> </complexContent>  
703 </complexType>
```

#### 704 **2.4.3.2.5 Element <ProxyRestrictionCondition>**

705 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent  
706 assertions of their own on the basis of the information contained in the original assertion. A relying party  
707 MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis  
708 of an assertion containing such a condition.

709 The <ProxyRestrictionCondition> element contains the following elements and attributes:

710 Count [Optional]

711 Specifies the number of indirections that MAY exist between this assertion and an assertion which  
712 has ultimately been issued on the basis of it.

713 <Audience> [Zero or More]

714 Specifies the set of audiences to whom new assertions MAY be issued on the basis of this assertion.

715 A Count value of zero indicates that a relying party MUST NOT issue an assertion to another relying  
716 party on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves  
717 contain a <ProxyRestrictionCondition> element with a Count value of at most one less than this  
718 value.

719 If no <Audience> elements are specified, then no restrictions are made upon the relying parties to  
720 whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST themselves  
721 contain an <AudienceRestrictionCondition> element with at least one of the <Audience>  
722 elements present in the previous <ProxyRestrictionCondition> element, and no <Audience>  
723 elements present that were not in the previous <ProxyRestrictionCondition> element.

724 A SAML authority SHOULD NOT include more than one <ProxyRestrictionCondition> element  
725 within a <Conditions> element of an assertion. If multiple <ProxyRestrictionCondition>  
726 elements appear within a <Conditions> element, a relying party MUST treat the multiple elements as  
727 though a single <ProxyRestrictionCondition> element was specified, with a Count value equal to  
728 the lowest of any specified, and the set of <Audience> elements consisting of the union of the elements  
729 specified.

730 For the purposes of determining the validity of the <Conditions> element, the  
731 <ProxyRestrictionCondition> is considered to always be valid.

732 The following schema fragment defines the <ProxyRestrictionCondition> element and its  
733 **ProxyRestrictionConditionType** complex type:

```
734 <element name="ProxyRestrictionCondition"  
735 type="saml:ProxyRestrictionConditionType"/>  
736 <complexType name="ProxyRestrictionConditionType">  
737   <complexContent>  
738     <extension base="saml:ConditionAbstractType">  
739       <sequence>  
740         <element ref="saml:Audience" minOccurs="0"  
741 maxOccurs="unbounded"/>  
742       </sequence>  
743       <attribute name="Count" type="nonNegativeInteger"  
744 use="optional"/>  
745     </extension>  
746   </complexContent>  
747 </complexType>
```

### 748 2.4.3.3 Element <Advice>

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

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

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

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

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

```
761 <element name="Advice" type="saml:AdviceType"/>  
762 <complexType name="AdviceType">  
763   <choice minOccurs="0" maxOccurs="unbounded">  
764     <element ref="saml:AssertionIDReference"/>  
765     <element ref="saml:AssertionURIReference"/>  
766     <element ref="saml:Assertion"/>  
767     <any namespace="##other" processContents="lax"/>  
768   </choice>  
769 </complexType>
```

## 770 2.5 Statements

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

### 772 2.5.1 Element <Statement>

773 The <Statement> element is an extension point that allows other assertion-based applications to reuse  
774 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable  
775 only as the base of a derived type. This element has an optional attribute:

776 SessionIndex [Optional]

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

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

```
782 <element name="Statement" type="saml:StatementAbstractType"/>  
783 <complexType name="StatementAbstractType" abstract="true">  
784   <attribute name="SessionIndex" type="string" use="optional"/>  
785 </complexType>
```

### 786 **2.5.1.1 Elements <SubjectConfirmation>, <ConfirmationMethod>, and** 787 **<SubjectConfirmationData>**

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

790 <ConfirmationMethod> [Required]

791 A URI reference that identifies a protocol to be used to authenticate the subject. URI references  
792 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the  
793 SAML profiles specification [SAMLProf]. Additional methods may be added by defining new URIs and  
794 profiles or by private agreement.

795 <SubjectConfirmationData> [Optional]

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

797 <ds:KeyInfo> [Optional]

798 An XML Signature [XMLSig] element that identifies a cryptographic key.

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

```
802 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
803 <complexType name="SubjectConfirmationType">  
804   <sequence>  
805     <element ref="saml:ConfirmationMethod"/>  
806     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
807     <element ref="ds:KeyInfo" minOccurs="0"/>  
808   </sequence>  
809 </complexType>  
810 <element name="SubjectConfirmationData" type="anyType"/>  
811 <element name="ConfirmationMethod" type="anyURI"/>
```

### 812 **2.5.2 Element <AuthenticationStatement>**

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

817 AuthenticationMethod [Required]

818 A URI reference that specifies the type of authentication that took place. URI references identifying  
819 common authentication protocols are listed in Section Authentication Method Identifiers. A value of  
820 urn:oasis:names:tc:SAML:2.0:am:authncontext indicates that an <AuthnContext>  
821 element is included in the statement that describes further details of the authentication.

822 AuthenticationInstant [Required]

823 Specifies the time at which the authentication took place. The time value is encoded in UTC as  
824 described in Section Time Values.

825 <SubjectLocality> [Optional]

826 Specifies the DNS domain name and IP address for the system from which the subject was  
827 apparently authenticated.

828 <AuthnContext> [Optional]

829 The context used by the identity provider in the authentication event that yielded this statement.  
830 Contains an authentication context statement or a reference to one. Optionally contains a reference  
831 to an authentication context class. [See the Authentication Context specification \[SAMLAuthnCxt\] for](#)  
832 [a full description of authentication context class information.](#)

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

835 <AuthenticationStatement> elements MUST contain a SessionIndex value, conforming to the  
836 rules specified in section 2.5.1.

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

```
839 <element name="AuthenticationStatement"  
840         type="saml:AuthenticationStatementType"/>  
841 <complexType name="AuthenticationStatementType">  
842     <complexContent>  
843         <extension base="saml:StatementAbstractType">  
844             <sequence>  
845                 <element ref="saml:SubjectLocality" minOccurs="0"/>  
846                 <element ref="saml:AuthnContext" minOccurs="0"/>  
847             </sequence>  
848             <attribute name="AuthenticationMethod" type="anyURI"  
849 use="required"/>  
850             <attribute name="AuthenticationInstant" type="dateTime"  
851 use="required"/>  
852         </extension>  
853     </complexContent>  
854 </complexType>
```

### 855 2.5.2.1 Element <SubjectLocality>

856 The <SubjectLocality> element specifies the DNS domain name and IP address for the system  
857 from which the subject was authenticated. It has the following attributes:

858 IPAddress [Optional]

859 The IP address of the system from which the subject was authenticated.

860 DNSAddress [Optional]

861 The DNS address of the system from which the subject was authenticated.

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

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

```
866 <element name="SubjectLocality"  
867         type="saml:SubjectLocalityType"/>  
868 <complexType name="SubjectLocalityType">  
869     <attribute name="IPAddress" type="string" use="optional"/>  
870     <attribute name="DNSAddress" type="string" use="optional"/>  
871 </complexType>
```

## 872 2.5.2.2 Element <AuthnContext>

873 The <AuthnContext> element specifies the context of an authentication event with an optional context  
874 class URI followed by an authentication context statement or statement reference. It's complex  
875 **AuthnContextType** has the following elements:

876 <AuthnContextClassRef> [Optional]

877 A URI identifying an authentication context class that describes the authentication context statement  
878 that follows.

879 <AuthnContextStatement> or <AuthnContextStatementRef> [Required]

880 Either an authentication context statement, or a URI that identifies such a statement. The URI MAY  
881 directly resolve into an XML document containing the referenced statement.

882 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**  
883 complex type:

```
884 <element name="AuthnContext" type="saml:AuthnContextType"/>
885 <complexType name="AuthnContextType">
886   <sequence>
887     <element ref="saml:AuthnContextClassRef" minOccurs="0"/>
888     <choice>
889       <element ref="saml:AuthnContextStatement"/>
890       <element ref="saml:AuthnContextStatementRef"/>
891     </choice>
892   </sequence>
893 </complexType>
894 <element name="AuthnContextClassRef" type="anyURI"/>
895 <element name="AuthnContextStatementRef" type="anyURI"/>
896 <element name="AuthnContextStatement" type="anyType"/>
```

## 897 2.5.3 Element <AttributeStatement>

898 The <AttributeStatement> element describes a statement by the SAML authority asserting that the  
899 statement's subject is associated with the specified attributes. It is of type **AttributeStatementType**,  
900 which extends **StatementAbstractType** with the addition of the following element:

901 <Attribute> [One or More]

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

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

```
905 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
906 <complexType name="AttributeStatementType">
907   <complexContent>
908     <extension base="saml:StatementAbstractType">
909       <sequence>
910         <element ref="saml:Attribute"
911           maxOccurs="unbounded"/>
912       </sequence>
913     </extension>
914   </complexContent>
915 </complexType>
```

### 916 2.5.3.1 Elements <AttributeDesignator> and <Attribute>

917 The <AttributeDesignator> element identifies an attribute name within an attribute namespace. It  
918 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that attribute

919 values within a specific namespace be returned (see Section Element <AttributeQuery> for more  
920 information). The <AttributeDesignator> element contains the following XML attributes:

921 Name [Required]

922 The name of the attribute.

923 NameFormat [OptionalRequired]

924 A URI reference representing the classification of the attribute name for purposes of interpreting  
925 the name. See Section 7.x for some URI references that MAY be used as the value of the  
926 NameFormat attribute and their associated descriptions and processing rules. If no  
927 NameFormat value is provided, the identifier urn:oasis:names:tc:SAML:2.0:attrname-  
928 format:unspecified (see Section 7.x) is in effect.

929 ValueType [Optional]

930 A URI reference representing the datatype of the desired or supplied attribute. If no ValueType  
931 value is provided, the identifier urn:oasis:names:tc:saml:2.0:valuetype-  
932 format:appSpecificUnspecified (see Section 7.x) is in effect. Note that datatypes specified on the  
933 <AttributeValue> element using xsi:type have no SAML-defined relationship with  
934 ValueType. The ValueType setting (default or explicit) in an attribute query using the  
935 <AttributeDesignator> element MUST be exactly matched (in addition to other exact  
936 matches as described in Section x) in order for an attribute to be returned.

937 Arbitrary attributes

938 This complex type uses an <xsd:anyAttribute> extension point to allow for arbitrary XML  
939 attributes to be added to <AttributeDesignator> constructs without the need for an explicit  
940 schema extension. This allows additional fields to be added as needed to supply additional  
941 parameters to be used in an attribute query. SAML extensions MUST NOT add local (non-  
942 namespace-qualified) XML attributes to the AttributeType complex type or to any element bound  
943 to this type or a derivation of it; such attributes are reserved for future maintenance and  
944 enhancement of SAML itself.

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

```
947 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
948 <complexType name="AttributeDesignatorType">  
949   <attribute name="Name" type="string" use="required"/>  
950   <attribute name="NameFormat" type="anyURI" use="requiredOptional"/>  
951   <attribute name="ValueType" type="anyURI" use="optional"/>  
952   <anyAttribute/>  
953 </complexType>
```

954 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the  
955 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following  
956 element and attributes:

957 Source [Optional]

958 The source location or database from which the attribute came. Interpretation of the source  
959 information is application-specific.

960 <AttributeValue> [Any Number]

961 The value of the attribute. If an attribute contains more than one discrete value, it is  
962 RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute  
963 exists but has no value, then the <AttributeValue> element MUST be omitted. If more than one  
964 <AttributeValue> element is supplied for an attribute, and any of the elements have a datatype  
965 assigned through xsi:type, then all of the <AttributeValue> elements must have the identical

966 datatype assigned.

#### 967 Arbitrary attributes

968 This complex type inherits from **AttributeDesignatorType** the ability to allow for uses an  
969 `<xsd:anyAttribute>` extension point to add arbitrary XML attributes to be added to  
970 `<Attribute>` constructs without the need for an explicit schema extension. This allows  
971 additional fields to be added as needed to supply the context in which the attribute should be  
972 understood. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes  
973 to the `AttributeType` complex type or to any element bound to this type or a derivation of it; such  
974 attributes are reserved for future maintenance and enhancement of SAML itself.

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

```
976 <element name="Attribute" type="saml:AttributeType"/>
977 <complexType name="AttributeType">
978   <complexContent>
979     <extension base="saml:AttributeDesignatorType">
980       <sequence>
981         <element ref="saml:AttributeValue" minOccurs="0"
982 maxOccurs="unbounded"/>
983       </sequence>
984       <anyAttribute/>
985     </extension>
986   </complexContent>
987 </complexType>
```

#### 988 2.5.3.1.1 Element `<AttributeValue>`

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

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

995 **Note:** Specifying a datatype on `<AttributeValue>` using `xsi:type` will require the  
996 presence of the extension schema that defines the datatype in order for schema  
997 processing to proceed.

998 The following schema fragment defines the `<AttributeValue>` element:

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

#### 1000 2.5.4 Element `<AuthorizationDecisionStatement>`

1001 **Note:** The `<AuthorizationDecisionStatement>` feature has been frozen as of  
1002 SAML V2.0, with no future enhancements planned. Users who require additional  
1003 functionality may want to consider the eXtensible Access Control Markup Language  
1004 [XACML], which offers enhanced authorization decision features.

1005 The `<AuthorizationDecisionStatement>` element describes a statement by the SAML authority  
1006 asserting that a request for access by the statement's subject to the specified resource has resulted in  
1007 the specified authorization decision on the basis of some optionally specified evidence.

1008 The resource is identified by means of a URI reference. In order for the assertion to be interpreted  
1009 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference  
1010 in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different

1011 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
1012 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1013 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
1014 dependent. When a scheme uses elements of the common syntax, it will also use the common  
1015 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
1016 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where  
1017 the port is elided.

1018 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the  
1019 URI normalized form wherever possible as follows:

- 1020 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1021 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1022 Inconsistent URI reference interpretation can also result from differences between the URI reference  
1023 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
1024 employed to specify an access control policy language. The following security conditions should be  
1025 satisfied by the system which employs SAML assertions:

- 1026 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
1027 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
1028 part of the resource URI reference.
- 1029 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
1030 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
1031 gain access to a denied resource by creating such an equivalence.

1032 The `<AuthorizationDecisionStatement>` element is of type  
1033 **AuthorizationDecisionStatementType**, which extends **StatementAbstractType** with the addition of the  
1034 following elements (in order) and attributes:

1035 **Resource** [Required]

1036 A URI reference identifying the resource to which access authorization is sought. It is permitted for  
1037 this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the  
1038 start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1039 **Decision** [Required]

1040 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
1041 the **DecisionType** simple type.

1042 **<Action>** [One or more]

1043 The set of actions authorized to be performed on the specified resource.

1044 **<Evidence>** [Optional]

1045 A set of assertions that the SAML authority relied on in making the decision.

1046 The following schema fragment defines the `<AuthorizationDecisionStatement>` element and its  
1047 **AuthorizationDecisionStatementType** complex type:

```
1048 <element name="AuthorizationDecisionStatement"  
1049 type="saml:AuthorizationDecisionStatementType"/>  
1050 <complexType name="AuthorizationDecisionStatementType">  
1051 <complexContent>  
1052 <extension base="saml:StatementAbstractType">  
1053 <sequence>  
1054 <element ref="saml:Action" maxOccurs="unbounded"/>
```

```

1055         <element ref="saml:Evidence" minOccurs="0"/>
1056     </sequence>
1057     <attribute name="Resource" type="anyURI" use="required"/>
1058     <attribute name="Decision" type="saml:DecisionType"
1059 use="required"/>
1060     </extension>
1061 </complexContent>
1062 </complexType>

```

#### 1063 2.5.4.1 Element <Action>

1064 The <Action> element specifies an action on the specified resource for which permission is sought. It  
1065 has the following attribute and string-data content:

1066 Namespace [Optional]

1067 A URI reference representing the namespace in which the name of the specified action is to be  
1068 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwdc-  
1069 negotiation specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

1070 *string data* [Required]

1071 An action sought to be performed on the specified resource.

1072 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```

1073 <element name="Action" type="saml:ActionType"/>
1074 <complexType name="ActionType">
1075     <simpleContent>
1076         <extension base="string">
1077             <attribute name="Namespace" type="anyURI"/>
1078         </extension>
1079     </simpleContent>
1080 </complexType>

```

#### 1081 2.5.4.2 Element <Evidence>

1082 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on  
1083 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one  
1084 or more of the following elements:

1085 <AssertionIDReference> [Any number]

1086 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1087 <AssertionURIReference> [Any number]

1088 Specifies an assertion by reference to a URI.

1089 <Assertion> [Any number]

1090 Specifies an assertion by value.

1091 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party  
1092 and the SAML authority making the authorization decision. For example, in the case that the SAML  
1093 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use  
1094 that assertion as evidence in making its authorization decision without endorsing the <Evidence>  
1095 element's assertion as valid either to the relying party or any other third party.

1096 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```

1097 <element name="Evidence" type="saml:EvidenceType"/>
1098 <complexType name="EvidenceType">

```

```
1099     <choice maxOccurs="unbounded">
1100         <element ref="saml:AssertionIDReference"/>
1101         <element ref="saml:AssertionURIReference"/>
1102         <element ref="saml:Assertion"/>
1103     </choice>
1104 </complexType>
```

## 3 SAML Protocols

1105

1106 SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of  
1107 protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting  
1108 queries, assertions, and other messages using existing widely deployed transport protocols.

1109 Specific SAML request and response messages derive from common types. The requester sends an  
1110 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an  
1111 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1112



1114

Figure 1: SAML Request-Response Protocol

1115 The protocols defined by SAML achieve the following actions are as follows:

- 1116 • Returning one or more requested Assertions request (includes a direct request of the desired  
1117 assertions, as well as querying for assertions that meet particular criteria)
- 1118 • Performing Request for authentication on request and returning the corresponding assertion to be  
1119 performed
- 1120 • Request to Registering a federated name or terminating a federated name registration on request
- 1121 • Request to Retrieve a protocol message that has been requested by means of an artifact
- 1122 • Request to terminate a federated name registration
- 1123 • Request for Performing a near-simultaneous logout of a collection of related sessions ("single  
1124 logout") on request
- 1125 • Request Providing a name identifier mapping on request

### 3.1 Schema Header and Namespace Declarations

1126

1127 The following schema fragment defines the XML namespaces and other header information for the  
1128 protocol schema:

```
1129 <schema  
1130   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1131   xmlns="http://www.w3.org/2001/XMLSchema"  
1132   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1133   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1134   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1135   elementFormDefault="unqualified"  
1136   attributeFormDefault="unqualified"  
1137   blockDefault="substitution"  
1138   version="2.0">  
1139   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
1140     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>  
1141   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
1142     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
1143     schema.xsd"/>
```

```
1144     <annotation>
1145         <documentation>
1146             Document identifier: sstc-saml-schema-protocol-2.0
1147             Location: http://www.oasis-
1148 open.org/committees/documents.php?wg_abbrev=security
1149         </documentation>
1150     </annotation>
1151     ...
1152 </schema>
```

## 1153 3.2 Requests and Responses

1154 The following sections define the SAML constructs that underlie request and response messages.

### 1155 3.2.1 Complex Type RequestAbstractType

1156 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
1157 This type defines common attributes and elements that are associated with all SAML requests:

1158 RequestID [Required]

1159 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in  
1160 Section 1.2.3 for identifier uniqueness. The values of the RequestID attribute in a request and the  
1161 InResponseTo attribute in the corresponding response MUST match.

1162 MajorVersion [Required]

1163 The major version of this request. The identifier for the version of SAML defined in this specification  
1164 is 2. SAML versioning is discussed in Section SAML Versioning.

1165 MinorVersion [Required]

1166 The minor version of this request. The identifier for the version of SAML defined in this specification  
1167 is 0. SAML versioning is discussed in Section SAML Versioning.

1168 IssueInstant [Required]

1169 The time instant of issue of the request. The time value is encoded in UTC as described in Section  
1170 Time Values.

1171 Consent [Optional]

1172 Indicates whether or not consent has been obtained from a user in the sending this request.

1173 <Issuer> [Optional]

1174 Identifies the entity that generated the request message.

1175 <ds:Signature> [Optional]

1176 An XML Signature that authenticates the request, as described in Section SAML and XML Signature  
1177 Syntax and Processing.

1178 <RelayState> [Optional]

1179 This contains state information that MUST be relayed back in the associated response.

1180 <Extensions> [Optional]

1181 This contains optional protocol message extension elements that are agreed upon between the  
1182 communicating parties.

1183 **Note:** The <RespondWith> element has been removed from <Request> for V2.0 of  
1184 SAML.

1185 The following schema fragment defines the **RequestAbstractType** complex type:

```
1186 <complexType name="RequestAbstractType" abstract="true">  
1187   <sequence>  
1188     <element ref="saml:Issuer" minOccurs="0"/>  
1189     <element ref="ds:Signature" minOccurs="0"/>  
1190     <element ref="samlp:RelayState" minOccurs="0"/>  
1191     <element ref="samlp:Extensions" minOccurs="0"/>  
1192   </sequence>  
1193   <attribute name="RequestID" type="ID" use="required"/>  
1194   <attribute name="MajorVersion" type="integer" use="required"/>  
1195   <attribute name="MinorVersion" type="integer" use="required"/>  
1196   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1197   <attribute name="Consent" type="anyURI" use="optional"/>  
1198 </complexType>  
1199 <element name="Extensions" type="samlp:ExtensionsType"/>  
1200 <complexType name="ExtensionsType">  
1201   <sequence>  
1202     <any namespace="##anyother" processContents="lax"  
1203     maxOccurs="unbounded"/>  
1204   </sequence>  
1205 </complexType>
```

### 1206 3.2.1.1 Element <RelayState>

1207 SAML requests MAY contain a string-valued element containing state information that the requester  
1208 wishes the responder to include in the response. This is particularly useful with asynchronous bindings  
1209 of protocol messages, such as the encoding of messages in browser URLs. This data SHOULD be  
1210 integrity-protected by the requester and MAY have other protections placed on it by the requester, such  
1211 as confidentiality. The length of this value SHOULD be kept as short as possible because of limitations  
1212 of the bindings in which it may be needed.

1213 The following schema fragment defines the <RelayState> element:

```
1214 <element name="RelayState" type="string"/>
```

### 1215 3.2.2 Complex Type StatusResponseType

1216 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This  
1217 type defines common attributes and elements that are associated with all SAML responses:

1218 ResponseID [Required]

1219 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in  
1220 Section 1.2.3 for identifier uniqueness.

1221 InResponseTo [Optional]

1222 A reference to the identifier of the request to which the response corresponds, if any. If the response  
1223 is not generated in response to a request, or if the RequestID attribute value of a request cannot be  
1224 determined (because the request is malformed), then this attribute MUST NOT be present.  
1225 Otherwise, it MUST be present and its value MUST match the value of the corresponding  
1226 RequestID attribute value.

1227 MajorVersion [Required]

1228 The major version of this response. The identifier for the version of SAML defined in this  
1229 specification is 2. SAML versioning is discussed in Section SAML Versioning.

1230 MinorVersion [Required]

1231 The minor version of this response. The identifier for the version of SAML defined in this  
1232 specification is 0. SAML versioning is discussed in Section SAML Versioning.

1233 IssueInstant [Required]

1234 The time instant of issue of the response. The time value is encoded in UTC as described in Section  
1235 Time Values.

1236 Recipient [Optional]

1237 The intended recipient of this response. This is useful to prevent malicious forwarding of responses  
1238 to unintended recipients, a protection that is required by some use profiles. It is set by the generator  
1239 of the response to a URI reference that identifies the intended recipient. If present, the actual  
1240 recipient MUST check that the URI reference identifies the recipient or a resource managed by the  
1241 recipient. If it does not, the response MUST be discarded.

1242 <Issuer> [Optional]

1243 Identifies the entity that generated the response message.

1244 <ds:Signature> [Optional]

1245 An XML Signature that authenticates the response, as described in Section SAML and XML  
1246 Signature Syntax and Processing.

1247 <RelayState> [Optional]

1248 This contains state information from the associated request being relayed back in the response. It  
1249 MUST match the <RelayState> value in the associated request, if any.

1250 <Extensions> [Optional]

1251 This contains optional protocol message extension elements that are agreed upon between the  
1252 communicating parties.

1253 <Status> [Required]

1254 A code representing the status of the corresponding request.

1255 The following schema fragment defines the **StatusResponseType** complex type:

```
1256 <complexType name="StatusResponseType">  
1257   <sequence>  
1258     <element ref="saml:Issuer" minOccurs="0"/>  
1259     <element ref="ds:Signature" minOccurs="0"/>  
1260     <element ref="samlp:RelayState" minOccurs="0"/>  
1261     <element ref="samlp:Extensions" minOccurs="0"/>  
1262     <element ref="samlp:Status"/>  
1263   </sequence>  
1264   <attribute name="ResponseID" type="ID" use="required"/>  
1265   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1266   <attribute name="MajorVersion" type="integer" use="required"/>  
1267   <attribute name="MinorVersion" type="integer" use="required"/>  
1268   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1269   <attribute name="Recipient" type="anyURI" use="optional"/>  
1270 </complexType>
```

### 1271 3.2.2.1 Element <Status>

1272 The <Status> element contains the following elements:

1273 <StatusCode> [Required]

1274 A code representing the status of the corresponding request.

1275 <StatusMessage> [Optional]  
1276 A message which MAY be returned to an operator.  
1277 <StatusDetail> [Optional]  
1278 Additional information concerning an error condition.

1279 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1280 <element name="Status" type="samlp:StatusType"/>  
1281 <complexType name="StatusType">  
1282   <sequence>  
1283     <element ref="samlp:StatusCode"/>  
1284     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1285     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1286   </sequence>  
1287 </complexType>
```

### 1288 3.2.2.2 Element <StatusCode>

1289 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the  
1290 corresponding request. The <StatusCode> element has the following element and attribute:

1291 Value [Required]

1292 The status code value. This attribute contains an XML Schema QName; a namespace prefix MUST  
1293 be provided. The value of the topmost <StatusCode> element MUST be from the top-level list  
1294 provided in this section.

1295 <StatusCode> [Optional]

1296 A subordinate status code that provides more specific information on an error condition.

1297 The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The  
1298 local parts of these QNames are as follows:

1299 Success

1300 The request succeeded.

1301 VersionMismatch

1302 The SAML responder could not process the request because the version of the request message  
1303 was incorrect.

1304 Requester

1305 The request could not be performed due to an error on the part of the requester.

1306 Responder

1307 The request could not be performed due to an error on the part of the SAML responder or SAML  
1308 authority.

1309 VersionMismatch

1310 The SAML responder could not process the request because the version of the request message  
1311 was incorrect.

1312 The following second-level status codes are referenced at various places in the specification. Additional  
1313 second-level status codes MAY be defined in future versions of the SAML specification.

1314 FederationDoesNotExist

1315 The responding provider does not recognize the federated <NameIdentifier> in the request.

1316	<u>InvalidNameIDPolicy</u>
1317	<u>The responding provider does not support the specified name identifier format for the requested</u>
1318	<u>subject.</u>
1319	<u>NoAuthnContext</u>
1320	<u>The specified authentication context requirements cannot be met by the responder.</u>
1321	<u>NoAvailableIDP</u>
1322	<u>Used by an intermediary to indicate that none of the supported identity provider &lt;Loc&gt; elements in</u>
1323	<u>an &lt;IDPList&gt; can be resolved or that none of the supported identity providers are available.</u>
1324	<u>NoPassive</u>
1325	<u>Indicates the identity provider cannot authenticate the principal passively, as has been requested.</u>
1326	<u>NoSupportedIDP</u>
1327	<u>Used by an intermediary to indicate that none of the identity providers in an &lt;IDPList&gt; are</u>
1328	<u>supported by the intermediary.</u>
1329	<u>ProxyCountExceeded</u>
1330	<u>Indicates that an identity provider cannot authenticate the principal directly and is not permitted to</u>
1331	<u>proxy the request further.</u>
1332	<u>RequestDenied</u>
1333	<u>The SAML responder or SAML authority is able to process the request but has chosen not to</u>
1334	<u>respond. This status code MAY be used when there is concern about the security context of the</u>
1335	<u>request message or the sequence of request messages received from a particular requester.</u>
1336	<u>RequestUnsupported</u>
1337	<u>The SAML responder or SAML authority does not support the request.</u>
1338	<u>RequestVersionDeprecated</u>
1339	<u>The SAML responder can not process any requests with the protocol version specified in the</u>
1340	<u>request.</u>
1341	RequestVersionTooHigh
1342	The SAML responder cannot process the request because the protocol version specified in the
1343	request message is a major upgrade from the highest protocol version supported by the responder.
1344	RequestVersionTooLow
1345	The SAML responder cannot process the request because the protocol version specified in the
1346	request message is too low.
1347	<del>RequestVersionDeprecated</del>
1348	<del>The SAML responder can not process any requests with the protocol version specified in the</del>
1349	<del>request.</del>
1350	<u>ResourceNotRecognized</u>
1351	<u>The SAML authority does not wish to support resource-specific attribute queries, or the resource</u>
1352	<u>value provided in the request message is invalid or unrecognized.</u>
1353	TooManyResponses
1354	The response message would contain more elements than the SAML responder will return.

1355 | ~~FederationDoesNotExist~~

1356 | ~~The responding provider does not recognize the federated <NameIdentifier> in the request.~~

1357 | ~~RequestDenied~~

1358 | ~~The SAML responder or SAML authority is able to process the request but has chosen not to~~  
 1359 | ~~respond. This status code MAY be used when there is concern about the security context of the~~  
 1360 | ~~request message or the sequence of request messages received from a particular requester.~~

1361 | ~~RequestUnsupported~~

1362 | ~~The SAML responder or SAML authority does not support the request.~~

1363 | ~~ResourceNotRecognized~~

1364 | ~~The SAML authority does not wish to support resource-specific attribute queries, or the resource~~  
 1365 | ~~value provided in the request message is invalid or unrecognized.~~

1366 | UnknownPrincipal

1367 | The responding provider does not recognize the principal specified or implied by the request.

1368 | ~~NoSupportedIDP~~

1369 | ~~Used by an intermediary to indicate that none of the identity providers in an <IDPList> are~~  
 1370 | ~~supported by the intermediary.~~

1371 | ~~NoAuthnContext~~

1372 | ~~The specified authentication context requirements cannot be met by the responder.~~

1373 | ~~InvalidNameIDPolicy~~

1374 | ~~The responding provider does not support the specified name identifier format for the requested~~  
 1375 | ~~subject.~~

1376 | ~~NoPassive~~

1377 | ~~Indicates the identity provider cannot authenticate the principal passively, as has been requested.~~

1378 | ~~ProxyCountExceeded~~

1379 | ~~Indicates that an identity provider cannot authenticate the principal directly and is not permitted to~~  
 1380 | ~~proxy the request further.~~

1381 | ~~NoAvailableIDP~~

1382 | ~~Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in~~  
 1383 | ~~an <IDPList> can be resolved or that none of the supported identity providers are available.~~

1384 | SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT  
 1385 | define additional codes in the SAML assertion or protocol namespace.

1386 | The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value  
 1387 | attribute and have the above semantics only in that context.

1388 | The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex  
 1389 | type:

```

1390 | <element name="StatusCode" type="samlp:StatusCodeType"/>
1391 | <complexType name="StatusCodeType">
1392 |   <sequence>
1393 |     <element ref="samlp:StatusCode" minOccurs="0"/>
1394 |   </sequence>
1395 |   <attribute name="Value" type="QName" use="required"/>
1396 | </complexType>

```

### 1397 3.2.2.3 Element <StatusMessage>

1398 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1399 The following schema fragment defines the <StatusMessage> element:

```
1400 <element name="StatusMessage" type="string"/>
```

### 1401 3.2.2.4 Element <StatusDetail>

1402 The <StatusDetail> element MAY be used to specify additional information concerning an error  
1403 condition.

1404 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
1405 complex type:

```
1406 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1407 <complexType name="StatusDetailType">  
1408   <sequence>  
1409     <any namespace="##any" processContents="lax" minOccurs="0"  
1410     maxOccurs="unbounded"/>  
1411   </sequence>  
1412 </complexType>
```

## 1413 3.3 Assertion Query and Request Protocol

1414 This section defines messages and processing rules for requesting existing assertions by reference or  
1415 querying for assertions by subject and statement type.

### 1416 3.3.1 Element <AssertionIDRequest>

1417 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>  
1418 message can be used to request that the assertion(s) be returned in a <Response> message. The  
1419 <saml:AssertionIDReference> element is used to specify the assertion(s) to return. See Section  
1420 Element <AssertionIDReference> for more information on this element.

1421 The following schema fragment defines the <AssertionIDRequest> element:

```
1422 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>  
1423 <complexType name="AssertionIDRequestType">  
1424   <complexContent>  
1425     <extension base="samlp:RequestAbstractType">  
1426       <sequence>  
1427         <element ref="saml:AssertionIDReference"  
1428         maxOccurs="unbounded"/>  
1429       </sequence>  
1430     </extension>  
1431   </complexContent>  
1432 </complexType>
```

1433

### 1434 3.3.2 Queries

1435 The following sections define the SAML query request messages.

### 1436 3.3.2.1 Element <SubjectQuery>

1437 The <SubjectQuery> message element is an extension point that allows new SAML queries to be  
1438 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and  
1439 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the <Subject>  
1440 element and an optional `SessionIndex` attribute to **RequestAbstractType**.

1441 `SessionIndex` [Optional]

1442 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1443 containing subject statements do you have for this subject within the context of the supplied session  
1444 information?”

1445 If the `SessionIndex` attribute is present in any defined query, at least one element that extends  
1446 **StatementAbstractType** in the set of returned assertions MUST contain an `SessionIndex` attribute  
1447 that matches the `SessionIndex` attribute in the query. It is OPTIONAL for the complete set of all such  
1448 matching assertions to be returned in the response.

1449 The following schema fragment defines the <SubjectQuery> element and its  
1450 **SubjectQueryAbstractType** complex type:

```
1451 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1452 <complexType name="SubjectQueryAbstractType" abstract="true">  
1453   <complexContent>  
1454     <extension base="samlp:RequestAbstractType">  
1455       <sequence>  
1456         <element ref="saml:Subject"/>  
1457       </sequence>  
1458       <attribute name="SessionIndex" type="string"  
1459         use="optional"/>  
1460     </extension>  
1461   </complexContent>  
1462 </complexType>
```

### 1463 3.3.2.2 Element <AuthenticationQuery>

1464 The <AuthenticationQuery> message element is used to make the query “What assertions  
1465 containing authentication statements are available for this subject?” A successful <Response> will  
1466 contain one or more assertions containing authentication statements.

1467 The <AuthenticationQuery> message MUST NOT be used as a request for a new authentication  
1468 using credentials provided in the request. <AuthenticationQuery> is a request for statements about  
1469 authentication acts that have occurred in a previous interaction between the indicated subject and the  
1470 Authentication Authority.

1471 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the  
1472 addition of the following attribute:

1473 `AuthenticationMethod` [Optional]

1474 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1475 containing authentication statements do you have for this subject with the supplied authentication  
1476 method?”

1477 In response to an authentication query, a SAML authority returns assertions with authentication  
1478 statements as follows:

- 1479 • Rules given in Section for matching against the <Subject> element of the query identify the  
1480 assertions that may be returned.

- 1481 • If the `AuthenticationMethod` attribute is present in the query, at least one  
1482 `<AuthenticationStatement>` element in the set of returned assertions MUST contain an  
1483 `AuthenticationMethod` attribute that matches the `AuthenticationMethod` attribute in  
1484 the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in  
1485 the response.

1486 The following schema fragment defines the `<AuthenticationQuery>` element and its  
1487 **AuthenticationQueryType** complex type:

```
1488 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
1489 <complexType name="AuthenticationQueryType">
1490   <complexContent>
1491     <extension base="samlp:SubjectQueryAbstractType">
1492       <attribute name="AuthenticationMethod" type="anyURI"/>
1493     </extension>
1494   </complexContent>
1495 </complexType>
```

### 1496 3.3.2.3 Element `<AttributeQuery>`

1497 The `<AttributeQuery>` element is used to make the query "Return the requested attributes for this  
1498 subject." A successful response will be in the form of assertions containing attribute statements. This  
1499 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of  
1500 the following element and attribute:

1501 `Resource` [Optional]

1502 If present, specifies that the attribute query is being made in order to evaluate a specific access  
1503 request relating to the resource. The SAML authority MAY use the resource attribute to establish the  
1504 scope of the request. It is permitted for this attribute to have the value of the empty URI reference  
1505 (""), and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]  
1506 §4.2.

1507 If the resource attribute is specified and the SAML authority does not wish to support resource-  
1508 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the  
1509 Attribute Authority SHOULD respond with a top-level `<StatusCode>` value of `Responder` and a  
1510 second-level `<StatusCode>` value of `ResourceNotRecognized`.

1511 `<AttributeDesignator>` [Any Number]

1512 Each `<AttributeDesignator>` element specifies an attribute whose value is to be returned. If no  
1513 attributes are specified, it indicates that all attributes allowed by policy are requested.

1514 In response to an attribute query, a SAML authority returns assertions with attribute statements as  
1515 follows:

- 1516 • Rules given in Section for matching against the `<Subject>` element of the query identify the  
1517 assertions that may be returned.
- 1518 • If any `<AttributeDesignator>` elements are present in the query, they constrain the attribute  
1519 values returned, as noted above.
- 1520 • The SAML authority MAY take the `Resource` attribute into account in further constraining the values  
1521 returned, as noted above.
- 1522 • The attribute values returned MAY be constrained by application-specific policy considerations.

1523 The following schema fragment defines the `<AttributeQuery>` element and its **AttributeQueryType**  
1524 complex type:

```

1525 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1526 <complexType name="AttributeQueryType">
1527   <complexContent>
1528     <extension base="samlp:SubjectQueryAbstractType">
1529       <sequence>
1530         <element ref="saml:AttributeDesignator"
1531           minOccurs="0" maxOccurs="unbounded"/>
1532       </sequence>
1533       <attribute name="Resource" type="anyURI" use="optional"/>
1534     </extension>
1535   </complexContent>
1536 </complexType>

```

### 1537 3.3.2.4 Element <AuthorizationDecisionQuery>

1538 The <AuthorizationDecisionQuery> element is used to make the query “Should these actions on  
1539 this resource be allowed for this subject, given this evidence?” A successful response will be in the form  
1540 of assertions containing authorization decision statements.

1541 **Note:** The <AuthorizationDecisionQuery> feature has been frozen as of SAML  
1542 V2.0, with no future enhancements planned. Users who require additional functionality  
1543 may want to consider the eXtensible Access Control Markup Language [XACML], which  
1544 offers enhanced authorization decision features.

1545 This element is of type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType**  
1546 with the addition of the following elements and attribute:

1547 Resource [Required]

1548 A URI reference indicating the resource for which authorization is requested.

1549 <Action> [One or More]

1550 The actions for which authorization is requested.

1551 <Evidence> [Optional]

1552 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1553 In response to an authorization decision query, a SAML authority returns assertions with authorization  
1554 decision statements as follows:

- 1555 • Rules given in Section 3.3.4.1 for matching against the <Subject> element of the query identify the  
1556 assertions that may be returned.

1557 The following schema fragment defines the <AuthorizationDecisionQuery> element and its  
1558 **AuthorizationDecisionQueryType** complex type:

```

1559 <element name="AuthorizationDecisionQuery"
1560 type="samlp:AuthorizationDecisionQueryType"/>
1561 <complexType name="AuthorizationDecisionQueryType">
1562   <complexContent>
1563     <extension base="samlp:SubjectQueryAbstractType">
1564       <sequence>
1565         <element ref="saml:Action" maxOccurs="unbounded"/>
1566         <element ref="saml:Evidence" minOccurs="0"/>
1567       </sequence>
1568       <attribute name="Resource" type="anyURI" use="required"/>
1569     </extension>
1570   </complexContent>
1571 </complexType>

```

### 1572 3.3.3 Element <Response>

1573 The <Response> message element is used when a response consists of a list of zero or more  
1574 assertions that answer the request. It has the complex type **ResponseType**, which extends  
1575 **StatusResponseType** by adding the following element:

1576 <Assertion> [Any Number]

1577 Specifies an assertion by value. (See Section Element <Assertion> for more information.)

1578 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1579 <element name="Response" type="samlp:ResponseType"/>
1580 <complexType name="ResponseType">
1581   <complexContent>
1582     <extension base="samlp:StatusResponseType">
1583       <sequence>
1584         <element ref="saml:Assertion" minOccurs="0"
1585 maxOccurs="unbounded"/>
1586       </sequence>
1587     </extension>
1588   </complexContent>
1589 </complexType>
```

#### 1590 3.3.3.1 Processing Rules

1591 In response to a query message, every assertion returned by a SAML authority **MUST** contain a  
1592 <Subject> element that **strongly matches** the <Subject> element found in the query.

1593 A <Subject> element S1 strongly matches S2 if and only if the following two conditions both apply:

- 1594 • If S2 includes an identifier element (any element whose type is derived from  
1595 **BaseIdentifierAbstractType**), then S1 must include an identical identifier element.
- 1596 • If S2 includes one or more <SubjectConfirmation> elements, then S1 must include at least one  
1597 <SubjectConfirmation> element such that the assertion's subject can be confirmed in the  
1598 manner described by at least one element in the requested set.

1599 If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
1600 expressed by a query, the <Response> element **MUST NOT** contain an <Assertion> element and  
1601 **MUST** include a <StatusCode> element with value *Success*. It **MAY** return a <StatusMessage>  
1602 element with additional information.

## 1603 3.4 Authentication Request Protocol

1604 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing  
1605 authentication statements to establish a security context at one or more relying parties, it can use the  
1606 authentication request protocol to send an <AuthnRequest> message to a SAML authority and request  
1607 that it return a <Response> message containing one or more such assertions. A SAML authority that  
1608 supports this protocol is also termed an identity provider. Such assertions **MAY** contain additional  
1609 statements of any type, but at least one assertion **MUST** contain at least one authentication statement.

1610 Apart from this requirement, the specific contents of the returned assertions depend on the profile or  
1611 context of use. Also, the exact means by which the principal or agent authenticates to the identity  
1612 provider are not specified, though the means of authentication **MAY** impact the content of the response.  
1613 Other issues related to the validation of authentication credentials by the identity provider or any  
1614 communication between the identity provider and any other entities involved in the authentication  
1615 process are also out of scope of this protocol.

1616 The descriptions and processing rules in the following sections reference the following actors, many of  
1617 whom might be the same entity in a particular profile of use:

1618 Request Issuer

1619 The entity who creates the authentication request and to whom the response is to be returned.

1620 Presenter

1621 The entity who presents the request to the authority and either authenticates itself during the  
1622 sending of the message, or relies on an existing security context to establish its identity. If not  
1623 the request issuer, the sender acts as an intermediary between the request issuer and the  
1624 responding identity provider.

1625 Requested Subject

1626 The entity about whom one or more assertions are being requested.

1627 Confirming Subject

1628 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`  
1629 elements of the resulting assertion(s).

1630 Relying Party

1631 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by  
1632 the profile or context of use, generally to establish a security context.

1633 **3.4.1 Element `<AuthnRequest>`**

1634 To request that an identity provider issue an authentication assertion, an entity authenticates to it (or  
1635 relies on an existing security context) and sends it an `<AuthnRequest>` message that describes the  
1636 properties that the resulting assertion needs to have to satisfy its purpose. Among these properties may  
1637 be information that relates to the content of the assertion and/or information that relates to how the  
1638 resulting `<Response>` message should be delivered to the request issuer.

1639 The request issuer might not be the same as the presenter of the request, if for example the request  
1640 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the  
1641 requested subject to provide a service.

1642 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected  
1643 by the protocol binding used to deliver the message.

1644 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and  
1645 adds the following elements and attributes, all of which are optional in general, but may be required by  
1646 specific profiles:

1647 `<Subject>` [Optional]

1648 Specifies the requested subject of the resulting assertion(s). This may include one or more  
1649 `<SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions' can  
1650 be confirmed.

1651 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the  
1652 requested subject. If no `<SubjectConfirmation>` elements are included, then the presenter is  
1653 presumed to be the only confirming entity required and the method is implied by the profile of use  
1654 and/or the policies of the identity provider.

1655 `<NameIDPolicy>` [Optional]

1656 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,

1657 then any type of identifier supported by the identity provider for the requested subject can be used,  
1658 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

1659

1660 <Conditions> [Optional]

1661 Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the  
1662 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary.

1663 <RequestAuthnContext> [Optional]

1664 Specifies the requirements, if any, that the request issuer places on the authentication context that  
1665 applies to the responding provider's authentication of the presenter.

1666 <Scoping> [Optional]

1667 Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as  
1668 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity  
1669 providers by the responder.

1670 IsPassive [Optional]

1671 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of  
1672 the user interface from the request issuer and interact with the presenter in a noticeable fashion. If a  
1673 value is not provided, the default is "true".

1674 ForceAuthn [Optional]

1675 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than  
1676 rely on a previous security context. If a value is not provided, the default is "false". However, if both  
1677 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the  
1678 presenter unless the constraints of IsPassive can be met.

1679 ProtocolBinding [Optional]

1680 A URI that identifies a SAML protocol binding to be used when returning the <Response> message.

1681 AssertionConsumerServiceID [Optional]

1682 References one of a set of <AssertionConsumerService> elements in the request issuer's  
1683 metadata as the one to which the <Response> should be returned. It applies only to profiles that  
1684 specify use of this metadata element, in which the request issuer is different than the presenter. If  
1685 omitted, the metadata element labeled with the isDefault attribute MUST be used with such  
1686 profiles.

1687 AssertionConsumerServiceURL [Optional]

1688 If the would-be presenter of an <AuthnRequest> recognizes that the issuer's request cannot be  
1689 satisfied for some reason, this attribute specifies where a <Response> message generated by that  
1690 would-be presenter MUST be returned. This attribute can be required by certain profiles.

1691 ProviderName [Optional]

1692 Specifies the human-readable name of the request issuer for use by the presenter's user agent or  
1693 the identity provider.

1694 See Section 3.4.1.8 for general processing rules regarding this message.

1695 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**  
1696 complex type:

```
1697 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>  
1698 <complexType name="AuthnRequestType">  
1699   <complexContent>
```

```

1700         <extension base="samlp:RequestAbstractType">
1701             <sequence>
1702                 <element ref="saml:Subject" minOccurs="0"/>
1703                 <element ref="samlp:NameIDPolicy" minOccurs="0"/>
1704                 <element
1705 ref="saml:Conditions" minOccurs="0"/>
1706                 <element ref="samlp:RequestAuthnContext"
1707 minOccurs="0"/>
1708                 <element ref="samlp:Scoping" minOccurs="0"/>
1709             </sequence>
1710             <attribute name="IsPassive" type="boolean"
1711 use="optional"/>
1712             <attribute name="ForceAuthn" type="boolean"
1713 use="optional"/>
1714             <attribute name="ProtocolBinding" type="anyURI"
1715 use="optional"/>
1716             <attribute name="AssertionConsumerServiceID" type="string"
1717 use="optional"/>
1718             <attribute name="AssertionConsumerServiceURL"
1719 type="anyURI" use="optional"/>
1720             <attribute name="ProviderName" type="string"
1721 use="optional"/>
1722         </extension>
1723     </complexContent>
1724 </complexType>

```

### 1725 3.4.1.1 Element <NameIDPolicy>

1726 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an  
1727 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

1728 **Format [Required]**

1729 Specifies the URI of a name identifier format defined in this or another specification (see Section 7.3  
1730 for examples).

1731 **SPNameQualifier [Optional]**

1732 Used with a **Format** of `urn:oasis:names:tc:SAML:2.0:nameid-format:federated` or  
1733 `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`, it optionally specifies that a  
1734 federated identifier be returned (or created) in the namespace of a service provider other than the  
1735 issuing service provider, or an affiliation group.

1736 When this element is used, if the content is not understood by or acceptable to the identity provider,  
1737 then a <Response> MUST be returned with a <Status> containing a second-level <StatusCode> of  
1738 `samlp:InvalidNameIDPolicy`.

1739 A **Format** of `urn:oasis:names:tc:SAML:2.0:nameid-format:federated` expresses the  
1740 request issuer's willingness, at the discretion of the requested subject, to establish an identity federation  
1741 for the subject with the identity provider, if one does not already exist. But note that when  
1742 <NameIDPolicy> is omitted, the identity provider MAY, at its (and the subject's) discretion, also  
1743 establish such an identity federation with the understanding that the issuing service provider might  
1744 ignore the federated and persistent aspect of the identifier.

1745 A **Format** of `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates that the  
1746 resulting assertion(s) MUST contain <EncryptedIdentifier> elements instead of plaintext. The  
1747 underlying name identifier's unencrypted form can be of any type supported by the identity provider for  
1748 the requested subject.

1749 Any **Format** value (or the omission of this element) MAY result in an <EncryptedIdentifier> in the  
1750 resulting assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

1751 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**  
1752 complex type:

```
1753 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
1754 <complexType name="NameIDPolicyType">
1755   <sequence/>
1756   <attribute name="Format" type="anyURI" use="required"/>
1757   <attribute name="SPNameQualifier" type="string" use="optional"/>
1758 </complexType>
```

### 1759 3.4.1.2 Element `<RequestAuthnContext>`

1760 The `<RequestAuthnContext>` element specifies the authentication context requirements of the  
1761 request issuer with respect to the authentication of the presenter. Its **RequestAuthnContextType**  
1762 complex type defines the following elements and attributes:

1763 `<AuthnContextClassRef>` or `<AuthnContextStatementRef>` [One or More]

1764 Specifies one or more URIs identifying authentication context classes or statements.

1765 Comparison [Optional]

1766 Specifies the comparison method used to evaluate the requested context classes or statements, one  
1767 of "exact", "minimum", "maximum", or "better". The default is "exact".

1768 If `<RequestAuthnContext>` is specified in an `<AuthnRequest>` message, the authentication  
1769 statement in the resulting assertion MUST contain an authentication context that conforms to the  
1770 requested context as described below.

1771 Either a set of class references or statement references can be used. Additionally, the set of supplied  
1772 references MUST be evaluated as an ordered set, where the first element is the most preferred  
1773 authentication context class or statement. If none of the specified classes or statements can be satisfied  
1774 in accordance with the rules below, then the identity provider MUST return a `<Response>` message with  
1775 a second-level `<StatusCode>` of `samlp:NoAuthnContext`.

1776 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication  
1777 statement MUST be the exact match of at least one of the authentication contexts specified.

1778 If Comparison is set to "minimum", then the resulting authentication context in the authentication  
1779 statement MUST be at least as strong (as deemed by the identity provider) as one of the authentication  
1780 contexts specified.

1781 If Comparison is set to "better", then the resulting authentication context in the authentication statement  
1782 MUST be stronger (as deemed by the identity provider) than any one of the authentication contexts  
1783 specified.

1784 If Comparison is set to "maximum", then the resulting authentication context in the authentication  
1785 statement MUST be as strong as possible (as deemed by the identity provider) without exceeding the  
1786 strength of at least one of the authentication contexts specified.

1787 The following schema fragment defines the `<RequestAuthnContext>` element and its  
1788 **RequestAuthnContextType** complex type:

```
1789 <element name="RequestAuthnContext" type="samlp:RequestAuthnContextType"/>
1790 <complexType name="RequestAuthnContextType">
1791   <choice>
1792     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1793     <element ref="saml:AuthnContextStatementRef"
1794 maxOccurs="unbounded"/>
1795   </choice>
1796
```

```

1797     <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1798 use="optional"/>
1799 </complexType>
1800 <simpleType name="AuthnContextComparisonType">
1801   <restriction base="string">
1802     <enumeration value="exact"/>
1803     <enumeration value="minimum"/>
1804     <enumeration value="maximum"/>
1805     <enumeration value="better"/>
1806   </restriction>
1807 </simpleType>

```

### 1808 3.4.1.3 Element <Scoping>

1809 The <Scoping> element specifies the identity providers trusted by the request issuer to authenticate the  
 1810 presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to  
 1811 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following  
 1812 elements and attribute:

1813 <IDPList> [Optional]

1814 An advisory list of identity providers and associated information that the request issuer deems  
 1815 acceptable to respond to the request.

1816 <RequesterID> [Zero or More]

1817 Identifies the set requesting entities on whose behalf the request issuer is acting. Used to  
 1818 communicate the chain of request issuers when proxying occurs, as described in section 3.4.1.9.

1819 ProxyCount [Optional]

1820 Specifies the number of proxying indirections permissible between the identity provider that receives  
 1821 this <AuthnRequest> and the identity provider who ultimately authenticates the principal. A count  
 1822 of zero permits no proxying, while omitting this attribute expresses no such restriction.

1823 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a  
 1824 <Response> message with a second-level <StatusCode> of samlp:NoAvailableIDP or  
 1825 samlp:NoSupportedIDP if it cannot contact or does not support any of the specified identity providers.

1826 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```

1827 <element name="Scoping" type="samlp:ScopingType"/>
1828 <complexType name="ScopingType">
1829   <sequence>
1830     <element ref="samlp:IDPList" minOccurs="0"/>
1831     <element ref="samlp:RequesterID" minOccurs="0"
1832 maxOccurs="unbounded"/>
1833   </sequence>
1834   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
1835 </complexType>
1836 <element name="RequesterID" type="anyURI"/>

```

### 1837 3.4.1.4 Element <IDPList>

1838 The <IDPList> element specifies the identity providers trusted by the request issuer to authenticate the  
 1839 presenter. Its **IDPListType** complex type defines the following elements:

1840 <IDPEntry> [One or More]

1841 Information about a single identity provider

1842 <GetComplete> [Optional]

1843 If the <IDPList> is not complete, this element may specify a URI that resolves to the complete list.

1844 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
1845 <element name="IDPList" type="samlp:IDPListType"/>
1846 <complexType name="IDPListType">
1847   <sequence>
1848     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
1849     <element ref="samlp:GetComplete" minOccurs="0"/>
1850   </sequence>
1851 </complexType>
1852 <element name="GetComplete" type="anyURI"/>
```

### 1853 3.4.1.5 Element <IDPEntry>

1854 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to  
1855 authenticate the presenter. Its **IDPEntryType** complex type defines the following elements:

1856 <ID> [Required]

1857 The unique identifier of the identity provider

1858 <Name> [Optional]

1859 A human readable name for the identity provider

1860 <Loc> [Optional]

1861 The location of a profile-specific endpoint supporting the authentication request protocol. The  
1862 binding to be used must be understood from the profile of use.

1863 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
1864 <element name="IDPEntry" type="samlp:IDPEntryType"/>
1865 <complexType name="IDPEntryType">
1866   <sequence/>
1867   <attribute name="ID" type="anyURI" use="required"/>
1868   <attribute name="Name" type="string" use="optional"/>
1869   <attribute name="Loc" type="anyURI" use="optional"/>
1870 </complexType>
```

### 1871 3.4.1.6 Processing Rules

1872 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is  
1873 therefore typically profiled for use in a specific context in which this optionality is constrained and specific  
1874 kinds of input and output are required or prohibited. The following processing rules apply as invariant  
1875 behavior across any profile of this protocol exchange.

1876 The recipient **MUST** validate any signature present on the request or response message. To be  
1877 considered valid, the signature provided **MUST** be the signature of the <Issuer> contained in the  
1878 message.

1879 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing  
1880 one or more assertions that meet the specifications defined by the request, or a <Status> describing  
1881 the error that occurred. The responder **MAY** conduct additional message exchanges with the request  
1882 sender as needed to initiate or complete the authentication process, subject to the nature of the protocol  
1883 binding and the authentication mechanism. As described in the next section, this includes proxying the  
1884 request by directing the presenter to another identity provider by issuing its own <AuthnRequest>  
1885 message, so that the resulting assertion can be used to authenticate the presenter to the original  
1886 responder.

1887 If the responder is unable to authenticate the presenter or does not recognize the requested subject, it  
1888 MUST return a <Response> with a <Status> containing a second-level <StatusCode> of  
1889 samlp:UnknownPrincipal.

1890 If the <Subject> element in the request is present, then the resulting assertions' <Subject> MUST  
1891 **strongly match** the request <Subject>, as described in section 3.3.4.1, except that the identifier MAY  
1892 be in a different form if specified by <NameIDPolicy>.

1893 All of the content defined specifically within <AuthnRequest> is optional, although some may be  
1894 required by certain profiles. In the absence of any specific content at all, the following behavior is  
1895 assumed:

- 1896 • The assertion(s) returned MUST contain a <Subject> element that represents the presenter.  
1897 The identifier type and format are determined by the identity provider. At least one statement  
1898 MUST be an <AuthenticationStatement> that describes the authentication performed by the  
1899 responder or authentication service associated with it.
- 1900 • The request presenter should, to the extent possible, be the only entity able to satisfy the  
1901 <SubjectConfirmation> of the assertion(s). In the case of weaker confirmation methods,  
1902 binding-specific or other mechanisms will be used to help satisfy this requirement.
- 1903 • The resulting assertion(s) MUST contain an <AudienceRestrictionCondition> element  
1904 referencing the request issuer as an acceptable relying party. Other audiences MAY be included  
1905 as deemed appropriate by the identity provider.
- 1906 •

### 1907 **3.4.1.7 Proxying**

1908 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or  
1909 cannot directly authenticate him/her, but believes that the presenter has already authenticated to another  
1910 identity provider, it may respond to the request by issuing a new <AuthnRequest> on its own behalf to  
1911 be presented to the other identity provider. The original identity provider is termed the proxying identity  
1912 provider.

1913 Upon the successful return of a <Response> to the proxying provider, the enclosed assertion MAY be  
1914 used to authenticate the presenter so that the proxying provider can issue an assertion of its own in  
1915 response to the original <AuthnRequest>, completing the overall message exchange. Both the  
1916 proxying and authenticating identity providers MAY include constraints on proxying activity in the  
1917 messages and assertions they issue, as described in previous sections, and below.

1918

1919 The request issuer can influence proxy behavior by including a <Scoping> element where the provider  
1920 sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be  
1921 proxied by including an ordered <IDPList> of preferred providers.

1922 An identity provider can control secondary use of its assertions by proxying identity providers using a  
1923 <ProxyRestrictionCondition> element in the assertions it issues.

#### 1924 **3.4.1.7.1 Processing Rules**

1925 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is  
1926 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY  
1927 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

1928 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity  
 1929 provider MUST return an error containing a second-level `<samlp:StatusCode>` value of  
 1930 `samlp:ProxyCountExceeded`, unless it can directly authenticate the presenter.

1931 If it chooses to proxy, when creating the new `<AuthnRequest>`, an identity provider MUST include  
 1932 equivalent or stricter forms of all the information included in the original request (such as authentication  
 1933 context policy). Note however that the proxying provider is free to specify whatever `<NameIDPolicy>` it  
 1934 wishes to maximize the chances of a successful response.

1935 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST  
 1936 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for  
 1937 example) will be honored by the authenticating provider.

1938 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less  
 1939 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new  
 1940 request SHOULD contain a `<ProxyCount>` attribute.

1941 If an `<IDPList>` was specified in the original request, the new request MUST also contain an  
 1942 `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the  
 1943 `<IDPList>`, but MUST NOT remove any from the list.

1944 The authentication request and response are processed in normal fashion, in accordance with the rules  
 1945 given in Section 3.4.1.8 and the profile of use. Once the presenter has authenticated to the proxying  
 1946 identity provider (by delivering a `<Response>`), the following steps are followed:

- 1947 • The proxying identity provider prepares a new assertion on its own behalf by copying in the  
 1948 relevant information from the original assertion. The original assertion will be restricted by  
 1949 `<AudienceRestrictionCondition>` to (at least) the proxying identity provider, while the new  
 1950 assertion's condition will reference (at least) the original request issuer.
- 1951 • The new assertion's `<Subject>` should contain an identifier that satisfies the original request  
 1952 issuer's preferences, as defined by its `<NameIDPolicy>` element.
- 1953 • The `<AuthenticationStatement>` in the new assertion MUST include an `<AuthnContext>`  
 1954 element containing an `<ac:AuthenticatingAuthority>` element referencing the identity  
 1955 provider to which the proxying identity provider referred the presenter. If the original assertion  
 1956 contains `<AuthnContext>` information that includes one or more  
 1957 `<ac:AuthenticatingAuthority>` elements, those elements SHOULD be included in the new  
 1958 assertion, with the new element placed after them.
- 1959 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider  
 1960 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be  
 1961 consistent over time across different requests. The value MUST not conflict with values used or  
 1962 generated by other SAML providers.
- 1963 • Any other `<AuthnContext>` information MAY be copied, translated, or omitted in accordance  
 1964 with the policies of the proxying identity provider, provided that the original requirements dictated  
 1965 by the request issuer are met.

1966 If, in the future, the identity provider is asked to authenticate the same presenter for a second request  
 1967 issuer, and this request is equally or less strict than the original request, the identity provider MAY skip  
 1968 the creation of a new `<AuthnRequest>` to the authenticating identity provider and immediately issue  
 1969 another assertion (assuming the original assertion it received is still valid). The concrete definition of  
 1970 "equally or less strict" is up to the proxying identity provider.

## 1971 3.5 Artifact Protocol

1972 The artifact protocol provides a mechanism by which SAML protocol messages can be transported in a  
1973 SAML binding by reference instead of by value. Both requests and responses can be obtained by  
1974 reference using this specialized protocol. A message sender, instead of binding a message to a transport  
1975 protocol, sends a small piece of data called an artifact using the binding. An artifact can take a variety of  
1976 forms, but must support a means by which the receiver can determine who sent it. If the receiver wishes,  
1977 it can then use this protocol in conjunction with a different (generally synchronous) SAML binding  
1978 protocol to dereference the artifact into the original protocol message. The most common use for this  
1979 mechanism is with bindings that cannot easily carry a message because of size constraints.

1980 Depending on the characteristics of the underlying message being passed by reference, the artifact  
1981 protocol MAY require protections such as mutual authentication, integrity protection, confidentiality, etc.  
1982 from the protocol binding used to dereference the artifact. In all cases, the artifact MUST exhibit a single-  
1983 use semantic such that once it has been successfully dereferenced, it can no longer be used by any  
1984 party.

1985 Regardless of the protocol message obtained, the result of dereferencing an artifact MUST be treated  
1986 exactly as if the message so obtained had been sent originally in place of the artifact.

### 1987 3.5.1 Element <ArtifactRequest>

1988 The <ArtifactRequest> message is used to request that a protocol message be returned in an  
1989 <ArtifactResponse> message by specifying an artifact that represents the protocol message. The  
1990 original transmission of the artifact is governed by the specific binding or profile of SAML that is being  
1991 used; see the SAML specifications for bindings [SAMLBind] and profiles [SAMLProf] for more information  
1992 on the use of artifacts in bindings and profiles.

1993 The <ArtifactRequest> message SHOULD be signed or otherwise authenticated and integrity  
1994 protected by the protocol binding used to deliver the message.

1995 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
1996 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

1997 This message has the complex type **ArtifactRequestType**, which extends **RequestAbstractType** and  
1998 adds the following element:

1999 <Artifact> [Required]

2000 The artifact value that the requester received and now wishes to translate into the protocol message  
2001 it represents. See [SAMLBind] for specific artifact format information.

2002 The following schema fragment defines the <ArtifactRequest> element and its  
2003 **ArtifactRequestType** complex type:

```
2004 <element name="ArtifactRequest" type="samlp:ArtifactRequestType"/>  
2005 <complexType name="ArtifactRequestType">  
2006   <complexContent>  
2007     <extension base="samlp:RequestAbstractType">  
2008       <sequence>  
2009         <element ref="samlp:Artifact"/>  
2010       </sequence>  
2011     </extension>  
2012   </complexContent>  
2013 </complexType>  
2014 <element name="Artifact" type="string"/>
```

## 2015 **3.5.2 Element <ArtifactResponse>**

2016 The recipient of an <ArtifactRequest> message MUST respond with an <ArtifactResponse>  
2017 message, which is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a  
2018 single optional wildcard element corresponding to the protocol message being returned. This wrapped  
2019 message element can be a request or a response.

2020 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity  
2021 protected by the protocol binding used to deliver the message.

2022 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2023 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2024 The following schema fragment defines the <ArtifactResponse> element and its  
2025 **ArtifactResponseType** complex type:

```
2026 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>  
2027 <complexType name="ArtifactResponseType">  
2028   <complexContent>  
2029     <extension base="samlp:StatusResponseType">  
2030       <sequence>  
2031         <any namespace="#any" processContents="lax"  
2032 minOccurs="0"/>  
2033       </sequence>  
2034     </extension>  
2035   </complexContent>  
2036 </complexType>
```

## 2037 **3.5.3 Processing Rules**

2038 The recipient MUST validate any signature present on the request or response message. To be  
2039 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2040 message.

2041 If the responder recognizes the artifact as valid, then it responds with the associated protocol message  
2042 in an <ArtifactResponse> message. Otherwise, it responds with an <ArtifactResponse>  
2043 message with no embedded message. In both cases, the <Status> element MUST include a  
2044 <StatusCode> element with the code value *Success*. A response message with no embedded  
2045 message inside it is termed an empty response in the remainder of this section.

2046 The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent  
2047 request with the same artifact by any requester results in an empty response as described above.

2048 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles,  
2049 MAY be intended for consumption by any party that receives it and can respond appropriately. In most  
2050 other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued  
2051 MUST be associated with the intended recipient of the message that the artifact represents. If the artifact  
2052 issuer receives an <ArtifactRequest> from a requester that cannot authenticate itself as the original  
2053 intended recipient, then the artifact issuer MUST return an empty response.

2054 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such  
2055 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and  
2056 return it in an <ArtifactRequest> to the issuer.

2057 Note that the <ArtifactResponse>'s InResponseTo attribute MUST contain the value of the  
2058 corresponding <AssertionRequest>'s RequestID attribute, but the embedded protocol message will  
2059 contain its own message identifier, and in the case of an embedded response, may contain a different  
2060 InResponseTo value that corresponds to the original request message to which the embedded  
2061 message is responding.

## 2062 3.6 Federated Name Registration Protocol

2063 When an identity provider and service provider first federate a principal's identity using a  
2064 `<NameIdentifier>` element with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
2065 `format:federated`, the identity provider generates an opaque value that serves as the initial name  
2066 identifier that both the service provider and the identity provider use in referring to the principal when  
2067 communicating with each other.

2068 Subsequent to federation, the service provider MAY register a different opaque value with the identity  
2069 provider. This opaque value is an attribute termed the `SPProvidedIdentifier`. Until the service provider  
2070 registers a different name, this attribute is omitted from `<NameIdentifier>` elements referring to the  
2071 principal.

2072 Either the service provider or the identity provider MAY register a new name identifier for a principal with  
2073 each other at any time following federation. The name identifiers specified by providers SHOULD be  
2074 unique across the identity providers with which the principal's identity is federated and SHOULD be  
2075 unique within the group of name identifiers that have been registered with the identity provider by this  
2076 service provider.

2077 Only federated identifiers (as defined by a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
2078 `format:federated`) can be replaced and set with this protocol; non-federated, encrypted, or transient  
2079 identifiers MUST NOT be used.

### 2080 3.6.1 Element `<RegisterNameIdentifierRequest>`

2081 To register an `SPProvidedIdentifier` attribute with an identity provider, the service provider sends a  
2082 `<RegisterNameIdentifierRequest>` message. The same message may be sent by an identity  
2083 provider, seeking to change the `<NameIdentifier>` value stored by the service provider.

2084 The `<RegisterNameIdentifierRequest>` message SHOULD be signed or otherwise authenticated  
2085 and integrity protected by the protocol binding used to deliver the message

2086 The `<Issuer>` of the request MUST contain the unique identifier of the requesting provider, with a  
2087 `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

2088 This message has the complex type **RegisterNameIdentifierRequestType**, which extends  
2089 **RequestAbstractType** and adds the following elements:

2090 `<NameIdentifier>` [Required]

2091 The federated name identifier and associated attributes that specify the principal as currently  
2092 recognized by the identity and service providers prior to this request.

2093 `<NewIdentifier>` [Required]

2094 The new federated identifier value to be used when communicating with the requesting provider  
2095 concerning this principal. If the requester is the service provider, the new identifier will appear in  
2096 subsequent `<NameIdentifier>` elements in the `SPProvidedIdentifier` attribute. If the  
2097 requester is the identity provider, the new value will appear in subsequent `<NameIdentifier>`  
2098 elements as the element's value.

2099 The following schema fragment defines the `<RegisterNameIdentifierRequest>` element and its  
2100 **RegisterNameIdentifierRequestType** complex type:

```
2101 <element name="NewIdentifier" type="string">  
2102 <element name="RegisterNameIdentifierRequest"  
2103 type="samlp:RegisterNameIdentifierRequestType"/>  
2104 <complexType name="RegisterNameIdentifierRequestType">  
2105 <complexContent>
```

```

2106         <extension base="samlp:RequestAbstractType">
2107             <sequence>
2108                 <element ref="saml:NameIdentifier"/>
2109                 <element ref="samlp:NewIdentifier"/>
2110             </sequence>
2111         </extension>
2112     </complexContent>
2113 </complexType>

```

### 2114 3.6.2 Element <RegisterNameIdentifierResponse>

2115 The recipient of a <RegisterNameIdentifierRequest> message MUST respond with a  
 2116 <RegisterNameIdentifierResponse> message, which is of type **StatusResponseType** with no  
 2117 additional content.

2118 The <RegisterNameIdentifierResponse> message SHOULD be signed or otherwise authenticated  
 2119 and integrity protected by the protocol binding used to deliver the message.

2120 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
 2121 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2122 The following schema fragment defines the <RegisterNameIdentifierResponse> element:

```

2123 <element name="RegisterNameIdentifierResponse"
2124         type="samlp:StatusResponseType"/>

```

### 2125 3.6.3 Processing Rules

2126 The recipient MUST validate any signature present on the request or response message. To be  
 2127 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
 2128 message.

2129 If the request includes a <NameIdentifier> for which no federation exists between the service  
 2130 provider and the identity provider, the responding provider MUST respond with a <Status> containing a  
 2131 second-level <StatusCode> of samlp:FederationDoesNotExist.

2132 If the service provider requests that its identifier be changed, the identity provider MUST include the  
 2133 <NewIdentifier> element's value as the SPProvidedIdentifier when subsequently  
 2134 communicating to the service provider regarding this principal.

2135 If the identity provider requests that its identifier be changed, the service provider MUST use the  
 2136 <NewIdentifier> element's value as the <NameIdentifier> element value when subsequently  
 2137 communicating with the identity provider regarding this principal.

2138 In either case, the <NameIdentifier> value in the request and its associated  
 2139 SPProvidedIdentifier attribute MUST contain the most recent name identifier information  
 2140 established between the providers for the principal. The NameQualifier attribute MUST contain the  
 2141 unique identifier of the identity provider. If the principal's identity federation is between the identity  
 2142 provider and an affiliation group of which the service provider is a member, then the SPNameQualifier  
 2143 attribute MUST contain the unique identifier of the affiliation group. Otherwise, it MUST contain the  
 2144 unique identifier of the service provider.

2145 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
 2146 systems at both the requester and the responder. Implementations might wish to allow each party to  
 2147 accept either identifier for some period of time following the successful completion of a name identifier  
 2148 change. Not doing so could result in the inability of the principal to access resources.

2149 All other processing rules associated with the underlying request and response messages MUST be  
2150 observed.

## 2151 3.7 Federation Termination Protocol

2152 When a principal (or an appropriate agent acting on his or her behalf) terminates an identity federation  
2153 between a service provider and an identity provider through an interaction with the service provider, the  
2154 service provider MUST send a <FederationTerminationNotification> message to the identity  
2155 provider. The service provider is stating that it will no longer accept authentication assertions from the  
2156 identity provider for the specified principal.

2157 Likewise, when a principal terminates an identity federation through an interaction with the identity  
2158 provider, the identity provider MUST send a <FederationTerminationNotification> message to  
2159 the service provider. In this case, the identity provider is stating that it will no longer provide  
2160 authentication assertions to the service provider for the specified principal.

2161 ~~Only federated identifiers (as defined by a Format of urn:oasis:names:tc:SAML:2.0:nameid-~~  
2162 ~~format:federated) can be replaced and set with this protocol; non-federated, encrypted, or transient~~  
2163 ~~identifiers MUST NOT be used.~~

### 2164 3.7.1 Element <FederationTerminationNotification>

2165 A provider sends a <FederationTerminationNotification> to the provider with which it is  
2166 terminating a federation. The <FederationTerminationNotification> message SHOULD be  
2167 signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the  
2168 message.

2169 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
2170 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2171 This message has the complex type **FederationTerminationNotificationType**, which extends  
2172 **RequestAbstractType** and adds the following elements:

2173 <NameIdentifier> [Required]

2174 The federated name identifier and associated attributes that specify the principal as currently  
2175 recognized by the identity and service providers prior to this request. Format MUST be  
2176 urn:oasis:names:tc:SAML:2.0:nameid-format:federated.

2177 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its  
2178 **RegisterNameIdentifierRequestType** complex type:

```
2179 <element name="FederationTerminationNotification"  
2180 type="samlp:FederationTerminationNotificationType"/>  
2181 <complexType name="FederationTerminationNotificationType">  
2182   <complexContent>  
2183     <extension base="samlp:RequestAbstractType">  
2184       <sequence>  
2185         <element ref="saml:NameIdentifier"/>  
2186       </sequence>  
2187     </extension>  
2188   </complexContent>  
2189 </complexType>
```

## 2190 **3.7.2 Element <FederationTerminationResponse>**

2191 The recipient of a <FederationTerminationNotification> message MUST respond with a  
2192 <FederationTerminationResponse> message, which is of type **StatusResponseType** with no  
2193 additional content.

2194 The <FederationTerminationResponse> message SHOULD be signed or otherwise authenticated  
2195 and integrity protected by the protocol binding used to deliver the message.

2196 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2197 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2198 The following schema fragment defines the <FederationTerminationResponse> element:

```
2199 <element name="FederationTerminationResponse"  
2200 type="samlp:StatusResponseType"/>
```

## 2201 **3.7.3 Processing Rules**

2202 The recipient MUST validate any signature present on the request or response message. To be  
2203 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2204 message.

2205 If the request includes a <NameIdentifier> for which no federation exists between the service  
2206 provider and the identity provider, the responding provider MUST respond with a <samlp:Status>  
2207 containing a second-level <samlp:StatusCode> of samlp:FederationDoesNotExist.

2208 Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been  
2209 terminated. A provider MAY choose to invalidate the session of a user for whom federation has been  
2210 terminated.

2211 All other processing rules associated with the underlying request and response messages MUST be  
2212 observed.

## 2213 **3.8 Single Logout Protocol**

2214 The single logout protocol provides a message exchange protocol by which all sessions provided by a  
2215 particular session authority are near-simultaneously terminated. The single logout protocol is used either  
2216 when a principal logs out at a session participant or when the principal logs out directly at the  
2217 session authority. This protocol may also be used to logout a principal due to a timeout. The reason for  
2218 the logout event may be indicated through the `reason` attribute.

2219 The principal may have established authenticated sessions both with the session authority, and  
2220 individual session participants, based on authentication assertions supplied by the session authority.

2221 When the principal invokes the single logout process at a session participant, the session participant  
2222 MUST send a <LogoutRequest> message to the session authority that provided the authentication  
2223 service related to that session at the session participant.

2224 When either the principal invokes a logout at the session authority, or a session participant sends a  
2225 logout request to the session authority specifying that principal, the session authority MUST send a  
2226 <LogoutRequest> message to each session participant to which it provided authentication assertions  
2227 under its current session with the principal, with the exception of the session participant that sent the  
2228 <LogoutRequest> message to the session authority.

### 2233 3.8.1 Element <LogoutRequest>

2234 A session participant or session authority sends a <LogoutRequest> message to indicate that a  
2235 session has been terminated.

2236 The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity  
2237 protected by the protocol binding used to deliver the message.

2238 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType**, and  
2239 adds the following elements and attributes:

2240 <NameIdentifier> [Required]

2241 The name identifier and associated attributes that specify the principal as currently recognized by  
2242 the identity and service providers prior to this request.

2243 <SessionIndex> [Optional]

2244 The identifier that indexes this session at the message recipient.

2245 NotOnOrAfter [Optional]

2246 The time at which the request expires.

2247 Reason [Optional]

2248 An indication of the reason for the logout, in the form of a URI reference.

2249 The following schema fragment defines the <LogoutRequest> element and associated  
2250 **LogoutRequestType** complex type:

```
2251 <element name="LogoutRequest" type="saml:LogoutRequestType"/>
2252 <complexType name="LogoutRequestType">
2253   <complexContent>
2254     <extension base="saml:RequestAbstractType">
2255       <sequence>
2256         <element ref="saml:NameIdentifier"/>
2257         <element nameRef="SessionIndex"/>type="string"
2258 minOccurs="0" maxOccurs="unbounded"/>
2259       </sequence>
2260       <attribute name="Reason" type="anyURIstring" minOccurs="0"/>
2261       <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
2262     </extension>
2263   </complexContent>
2264 </complexType>
2265 <element name="SessionIndex" type="string" minOccurs="0"
2266 maxOccurs="unbounded"/>
```

### 2267 3.8.2 Element <LogoutResponse>

2268 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message,  
2269 of type **StatusResponseType**, with no additional content specified.

2270 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity  
2271 protected by the protocol binding used to deliver the message.

2272 The following schema fragment defines the <LogoutResponse> element:

```
2273 <element name="LogoutResponse" type="saml:StatusResponseType"/>
```

### 2274 3.8.3 Processing Rules

2275 The <Issuer> of either message in this protocol MUST contain the unique identifier of the requesting or  
2276 responding provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-  
2277 format:provider.

2278 Message recipients MUST validate any signature present on the messages specified in this protocol. To  
2279 be considered valid, the signature provided must be the signature of the <Issuer> contained in the  
2280 message.

2281 The message sender MAY use the Reason attribute to indicate the reason for sending the  
2282 <LogoutRequest>. Other values MAY be agreed upon between participants, but the following values  
2283 are defined directly by this specification for use by all message senders:

2284 urn:oasis:names:tc:SAML:2.0:logout:user

2285 Specifies that the message is being sent because the principal wishes to terminate the indicated  
2286 session.

2287 urn:oasis:names:tc:SAML:2.0:logout:admin

2288 Specifies that the message is being sent because an administrator wishes to terminate the indicated  
2289 session for that principal.

2290 All other processing rules associated with the underlying request and response messages MUST be  
2291 observed.

#### 2292 3.8.3.1 Session Participant Rules

2293 When a session participant receives a <LogoutRequest>, the session participant MUST authenticate  
2294 the message.. If the sender is the authority that provided an assertion linked to the principal's current  
2295 session, the session participant MUST invalidate the principal's session(s) referred to by the  
2296 <NameIdentifier> element, and any <SessionIndex> elements supplied in the message.

2297  
2298 The session participant MUST apply the logout request message to any assertion that meets the  
2299 following conditions, even if the assertion arrives after the logout request:

- 2300 • The <SessionIndex> of the assertion's statements matches one specified in the logout request.
- 2301 • The assertion would otherwise be valid
- 2302 • The logout request has not yet expired (determined by examining the NotOnOrAfter attribute on  
2303 the message).

#### 2304 3.8.3.2 Session Authority Rules

2305 When a session authority receives a <LogoutRequest>, the session authority MUST authenticate the  
2306 sender. If the sender is a session participant to which the session authority provided an assertion for the  
2307 current session, then the session authority SHOULD do the following:

- 2308 • Send a <LogoutRequest> message to each session participant for which the session authority  
2309 provided assertions in the current session, *other than the originator of a current*  
2310 *<LogoutRequest>*.
- 2311 • Send a <LogoutRequest> message to any session authority on behalf of whom the session  
2312 authority proxied the user's authentication, unless the second authority is the originator of the  
2313 <LogoutRequest>.

- 2314 • Terminate the principal's current session as specified by the `<NameIdentifier>` element, and  
2315 any `<SessionIndex>` elements present in the logout request message.
- 2316 It should be noted that a session authority MAY initiate a logout for reasons other than having received a  
2317 `<LogoutRequest>` from a session participant – these include, but are not limited to:
- 2318 • If some timeout period was agreed out-of-band with an individual session participant, the session  
2319 authority MAY send a `<LogoutRequest>` to that individual participant alone.
  - 2320 • An agreed global timeout period has been exceeded.
  - 2321 • The principal, or some other trusted entity has requested logout of the principal, directly at the  
2322 session authority.
  - 2323 • The session authority has determined that the principal's credentials may have been compromised.
- 2324 When constructing a logout request message, the session authority MUST set the value of the  
2325 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message.
- 2326 In addition to the values specified in section 3.6.3 for the `Reason` attribute, the following values are also  
2327 available for use by the session authority only:
- 2328 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`
- 2329 Specifies that the message is being sent because of the global session timeout interval period  
2330 being exceeded.
- 2331 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`
- 2332 Specifies that the message is being sent because a timeout interval period agreed between a  
2333 participant and the authority has been exceeded.
- 2334 If an error occurs during this further processing of the logout (for example, relying session participants  
2335 may not all implement the particular single logout protocol binding used by the requesting session  
2336 participant), then the session authority MUST respond to the original requester with a  
2337 `<LogoutResponse>` message, indicating the status of the logout request. The value  
2338 `samlp:UnsupportedBinding` is provided for a second-level `<samlp:StatusCode>`, indicating that a  
2339 session participant should retry the `<LogoutRequest>` using a different protocol binding.

## 2340 **3.9 Name Identifier Mapping Protocol**

- 2341 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name  
2342 identifier for the same principal in a particular format or federation namespace, it can send a request to  
2343 the identity provider using this protocol.
- 2344 For example, a service provider that wishes to communicate with another service provider with whom it  
2345 does not share an identity federation for the principal can use an identity provider that shares an identity  
2346 federation for the principal with both service providers to map from its own federated identifier to a new  
2347 identifier, generally encrypted, with which it can communicate with the second service provider.
- 2348 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into an  
2349 `<EncryptedIdentifier>` element unless a specific deployment dictates such protection is  
2350 unnecessary.

### 2351 **3.9.1 Element `<NameIdentifierMappingRequest>`**

- 2352 To request an alternate name identifier for a principal from an identity provider, a requester sends an  
2353 `<NameIdentifierMappingRequest>` message. This message has the complex type

2354 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following  
2355 element:

2356 <BaseIdentifier> or <NameIdentifier> or <EncryptedIdentifier> [Required]

2357 The identifier and associated attributes that specify the principal as currently recognized by the  
2358 requester and the responder.

2359 <NameIDPolicy>

2360 The format and optional name qualifier that describes the requirements for the identifier to be  
2361 returned.

2362 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2363 binding used to deliver the message.

2364 The following schema fragment defines the <NameIdentifierMappingRequest> element and its  
2365 **NameIdentifierMappingRequestType** complex type:

```
2366 <element name="NameIdentifierMappingRequest"  
2367 type="samlp:NameIdentifierMappingRequestType"/>  
2368 <complexType name="NameIdentifierMappingRequestType">  
2369 <complexContent>  
2370 <extension base="samlp:RequestAbstractType">  
2371 <sequence>  
2372 <choice>  
2373 <element ref="saml:BaseIdentifier"/>  
2374 <element ref="saml:NameIdentifier"/>  
2375 <element ref="saml:EncryptedIdentifier"/>  
2376 </choice>  
2377 <element ref="samlp:NameIDPolicy"/>  
2378 </sequence>  
2379 </extension>  
2380 </complexContent>  
2381 </complexType>
```

### 2382 3.9.2 Element <NameIdentifierMappingResponse>

2383 The recipient of a <NameIdentifierMappingRequest> message MUST respond with a  
2384 <NameIdentifierMappingResponse> message. This message has the complex type  
2385 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following  
2386 element:

2387 <NameIdentifier> or <EncryptedIdentifier> [Required]

2388 The identifier and associated attributes that specify the principal in the manner requested, usually in  
2389 encrypted form.

2390 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2391 binding used to deliver the message.

2392 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2393 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2394 The following schema fragment defines the <NameIdentifierMappingResponse> element and its  
2395 **NameIdentifierMappingResponseType** complex type:

```
2396 <element name="NameIdentifierMappingResponse"  
2397 type="samlp:NameIdentifierMappingResponseType"/>  
2398 <complexType name="NameIdentifierMappingResponseType">  
2399 <complexContent>  
2400 <extension base="samlp:StatusResponseType">  
2401 <choice>
```

```
2402         <element ref="saml:NameIdentifier">
2403         <element ref="saml:EncryptedIdentifier">
2404             </choice>
2405         </extension>
2406     </complexContent>
2407 </complexType>
```

### 2408 3.9.3 Processing Rules

2409 The recipient **MUST** validate any signature present on the request or response message. To be  
2410 considered valid, the signature provided **MUST** be the signature of the <Issuer> contained in the  
2411 message.

2412 If the responder does not recognize the principal identified in the request, it **MUST** respond with a  
2413 <Status> containing a second-level <StatusCode> of `samlp:UnknownPrincipal`.

2414 At the responder's discretion, the `samlp:InvalidNameIDPolicy` status code **MAY** be returned to  
2415 indicate an inability or unwillingness to supply an identifier in the requested format. Likewise, the  
2416 `samlp:FederationDoesNotExist` status code **MAY** be used to indicate that a requested federated  
2417 identifier cannot be returned.

2418 All other processing rules associated with the underlying request and response messages **MUST** be  
2419 observed.

---

## 2420 4 SAML Versioning

2421 The SAML specification set is versioned in two independent ways. Each is discussed in the following  
2422 sections, along with processing rules for detecting and handling version differences, when applicable.  
2423 Also included are guidelines on when and why specific version information is expected to change in  
2424 future revisions of the specification.

2425 When version information is expressed as both a Major and Minor version, it may be expressed  
2426 discretely, or in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version  
2427 number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

2428  $Major_B > Major_A \vee ( ( Major_B = Major_A ) \wedge Minor_B > Minor_A )$

### 2429 4.1 SAML Specification Set Version

2430 Each release of the SAML specification set will contain a major and minor version designation describing  
2431 its relationship to earlier and later versions of the specification set. The version will be expressed in the  
2432 content and filenames of published materials, including the specification set document(s), and XML  
2433 schema instance(s). There are no normative processing rules surrounding specification set versioning,  
2434 since it merely encompasses the collective release of normative specification documents which  
2435 themselves contain processing rules.

2436 The overall size and scope of changes to the specification set document(s) will informally dictate whether  
2437 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards  
2438 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics  
2439 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a  
2440 major revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0,  
2441 described in Section .

#### 2442 4.1.1 Schema Version

2443 As a non-normative documentation mechanism, any XML schema instances published as part of the  
2444 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the  
2445 specification set version in which it has been published. Validating implementations MAY use the  
2446 attribute as a means of distinguishing which version of a schema is being used to validate messages, or  
2447 to support a multiplicity of versions of the same logical schema.

#### 2448 4.1.2 SAML Assertion Version

2449 The SAML <Assertion> element contains attributes for expressing the major and minor version of the  
2450 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to  
2451 document the syntax, semantics, and processing rules of the assertions of the same version. That is,  
2452 specification set version 1.0 describes assertion version 1.0, and so on.

2453 There is explicitly NO relationship between the assertion version and the SAML assertion XML  
2454 namespace that contains the schema definitions for that assertion version.

2455 The following processing rules apply:

- 2456 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported  
2457 by the authority.
- 2458 • A SAML relying party MUST NOT process any assertion with a major assertion version number not  
2459 supported by the relying party.

- 2460 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version  
2461 number is higher than the minor assertion version number supported by the relying party. However,  
2462 all assertions that share a major assertion version number MUST share the same general processing  
2463 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1  
2464 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0  
2465 assertion without ill effect.

### 2466 4.1.3 SAML Protocol Version

2467 The SAML protocol <Request> and <Response> elements contain attributes for expressing the major  
2468 and minor version of the request or response message using a pair of integers. Each version of the  
2469 SAML specification set will be construed so as to document the syntax, semantics, and processing rules  
2470 of the protocol messages of the same version. That is, specification set version 1.0 describes request  
2471 and response version V1.0, and so on.

2472 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace  
2473 that contains the schema definitions for protocol messages for that protocol version.

2474 The version numbers used in SAML protocol <Request> and <Response> elements will be the same  
2475 for any particular revision of the SAML specification set.

#### 2476 4.1.3.1 Request Version

2477 The following processing rules apply to requests:

- 2478 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
2479 SAML requester and the SAML responder.
- 2480 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume  
2481 that it supports requests with the highest request version supported by the requester.
- 2482 • A SAML requester MUST NOT issue a request message with a request version number matching a  
2483 response version number that the requester does not support.
- 2484 • A SAML responder MUST reject any request with a major request version number not supported by  
2485 the responder.
- 2486 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
2487 higher than the highest supported request version that it supports. However, all requests that share a  
2488 major request version number MUST share the same general processing rules and semantics, and  
2489 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax  
2490 of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

### 2491 4.1.4 Response Version

2492 The following processing rules apply to responses:

- 2493 • A SAML responder MUST NOT issue a response message with a response version number higher  
2494 than the request version number of the corresponding request message.
- 2495 • A SAML responder MUST NOT issue a response message with a major response version number  
2496 lower than the major request version number of the corresponding request message except to report  
2497 the error `RequestVersionTooHigh`.

2498 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-  
2499 level <StatusCode> value of `VersionMismatch`, and MAY result in reporting one of the following

2500 **second-level values:** RequestVersionTooHigh, RequestVersionTooLow, or  
2501 RequestVersionDeprecated.

## 2502 **4.1.5 Permissible Version Combinations**

2503 In general, assertions of a particular major version may appear in response messages of the same major  
2504 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol  
2505 schema. Future versions of this specification are expected to explicitly describe the permitted  
2506 combinations across major versions.

2507 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to  
2508 appear in a V1.1 response message.

## 2509 **4.2 SAML Namespace Version**

2510 XML schema instances and "qualified names" (QNames) published as part of the specification set  
2511 contain one or more target namespaces into which the type, element, and attribute definitions are  
2512 placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and  
2513 syntactical definitions that make up that part of the specification.

2514 The namespace URIs defined by the specification set will generally contain version information of the  
2515 form *Major.Minor* somewhere in the URI. The major and minor version in the URI **MUST** correspond to  
2516 the major and minor version of the specification set in which the namespace is first introduced and  
2517 defined. This information is not typically consumed by an XML processor, which treats the namespace  
2518 opaquely, but is intended to communicate the relationship between the specification set and the  
2519 namespaces it defines.

2520 As a general rule, implementers can expect the namespaces (and the associated schema definitions)  
2521 defined by a major revision of the specification set to remain valid and stable across minor revisions of  
2522 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,  
2523 but this is expected to be rare. In such cases, the older namespaces and their associated definitions  
2524 should be expected to remain valid until a major specification set revision.

### 2525 **4.2.1 Schema Evolution**

2526 In general, maintaining namespace stability while adding or changing the content of a schema are  
2527 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
2528 older implementations will react to any given change, making forward compatibility difficult to achieve.  
2529 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of  
2530 namespace stability. Except in special circumstances (for example to correct major deficiencies or fix  
2531 errors), implementations should expect forward compatible schema changes in minor revisions, allowing  
2532 new messages to validate against older schemas.

2533 Implementations **SHOULD** expect and be prepared to deal with new extensions and message types in  
2534 accordance with the processing rules laid out for those types. Minor revisions **MAY** introduce new types  
2535 that leverage the extension facilities described in Section SAML Extensions. Older implementations  
2536 **SHOULD** reject such extensions gracefully when they are encountered in contexts that dictate mandatory  
2537 semantics. Examples include new query, statement, or condition types.

2538

## 5 SAML and XML Signature Syntax and Processing

2539 SAML assertions and SAML protocol request and response messages may be signed, with the following  
2540 benefits:

- 2541 • An assertion signed by the SAML authority supports:
  - 2542 – Assertion integrity.
  - 2543 – Authentication of the SAML authority to a SAML relying party.
  - 2544 – If the signature is based on the SAML authority's public-private key pair, then it also provides for  
2545 non-repudiation of origin.
- 2546 • A SAML protocol request or response message signed by the message originator supports:
  - 2547 – Message integrity.
  - 2548 – Authentication of message origin to a destination.
  - 2549 – If the signature is based on the originator's public-private key pair, then it also provides for non-  
2550 repudiation of origin.

2551 A digital signature is not always required in SAML. For example, it may not be required in the following  
2552 situations:

- 2553 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains  
2554 protection from a signature on the containing protocol response message. "Inherited" signatures  
2555 should be used with care when the contained object (such as the assertion) is intended to have a  
2556 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,  
2557 exposing the XML content and adding potentially unnecessary overhead.
- 2558 • The SAML relying party or SAML requester may have obtained an assertion or protocol message  
2559 from the SAML authority or SAML responder directly (with no intermediaries) through a secure  
2560 channel, with the SAML authority or SAML responder having authenticated to the relying party or  
2561 SAML responder by some means other than a digital signature.

2562 Many different techniques are available for "direct" authentication and secure channel establishment  
2563 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
2564 the applicable security requirements depend on the communicating applications and the nature of the  
2565 assertion or message transported.

2566 It is recommended that, in all other contexts, digital signatures be used for assertions and request and  
2567 response messages. Specifically:

- 2568 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority  
2569 SHOULD be signed by the SAML authority.
- 2570 • A SAML protocol message arriving at a destination from an entity other than the originating site  
2571 SHOULD be signed by the origin site.

2572 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that  
2573 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures  
2574 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure  
2575 compatibility with profiles that may require other mechanisms.

2576 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST  
2577 be used if signing.

## 2578 **5.1 Signing Assertions**

2579 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema  
2580 as described in Section Assertions.

## 2581 **5.2 Request/Response Signing**

2582 All SAML protocol request and response messages MAY be signed using the XML Signature. This is  
2583 reflected in the schema as described in Sections Requests and Responses and .

## 2584 **5.3 Signature Inheritance**

2585 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`  
2586 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a  
2587 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a  
2588 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,  
2589 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
2590 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
2591 and signature options.

2592 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
2593 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may  
2594 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
2595 information from the surrounding context, but no such inheritance should be inferred unless specifically  
2596 identified by the profile.

## 2597 **5.4 XML Signature Profile**

2598 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
2599 and many choices. This section details the constraints on these facilities so that SAML processors do not  
2600 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
2601 **xsd:ID**-typed attributes optionally present on the root elements to which signatures can apply: the  
2602 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the  
2603 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section  
2604 as the identifier attributes.

### 2605 **5.4.1 Signing Formats and Algorithms**

2606 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
2607 detached.

2608 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
2609 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
2610 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 2611 **5.4.2 References**

2612 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the  
2613 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element  
2614 may or may not be the root element of the actual XML document containing the signed assertion or  
2615 message.

2616 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier  
2617 attribute value of the root element of the message being signed. For example, if the attribute value is  
2618 "foo", then the URI attribute in the `<ds:Reference>` element MUST be "#foo".

### 2619 **5.4.3 Canonicalization Method**

2620 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the  
2621 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`  
2622 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages  
2623 embedded in an XML context can be verified independent of that context.

### 2624 **5.4.4 Transforms**

2625 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
2626 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive  
2627 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
2628 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2629 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
2630 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This  
2631 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
2632 applying the transforms manually to the content and reverifying the result as consisting of the same  
2633 SAML message.

### 2634 **5.4.5 KeyInfo**

2635 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
2636 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
2637 be absent.

### 2638 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

2639 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section  
2640 SAML Assertions.

### 2641 **5.4.7**

### 2642 **5.4.8 Example**

2643 Following is an example of a signed response containing a signed assertion. Line breaks have been  
2644 added for readability; the signatures are not valid and cannot be successfully verified.

```
2645 <Response  
2646   IssueInstant="2003-04-17T00:46:02Z"  
2647   MajorVersion="1"  
2648   MinorVersion="1"  
2649   Recipient="www.opensaml.org"  
2650   ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"  
2651   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
2652   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
2653   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2654   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
2655 <ds:Signature  
2656   xmlns:ds="http://www.w3.org/2000/09/xmlsig#">
```



```

2724 <NameIdentifier
2725   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
2726   scott@example.org</NameIdentifier>
2727 <SubjectConfirmation>
2728 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
2729 </SubjectConfirmation></Subject>
2730 <SubjectLocality
2731   IPAddress="127.0.0.1"/>
2732 </AuthenticationStatement>
2733 <ds:Signature
2734   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2735   <ds:SignedInfo>
2736     <ds:CanonicalizationMethod
2737       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2738     <ds:SignatureMethod
2739       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2740     <ds:Reference
2741       URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
2742       <ds:Transforms>
2743         <ds:Transform
2744           Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
2745         <ds:Transform
2746           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2747         <InclusiveNamespaces
2748           PrefixList="#default saml samlp ds xsd xsi"
2749           xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2750       </ds:Transform>
2751     </ds:Transforms>
2752     <ds:DigestMethod
2753       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2754     <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
2755     </ds:Reference>
2756   </ds:SignedInfo>
2757   <ds:SignatureValue>
2758     hq4zk+ZknjggCQgZm7ea8fI79gJESRy3E8LHDpYXWQIgzpkJN9CMLG8ENR4Nrw+n
2759     7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmt3TD
2760     MWuL/cBUj20tBZOQMF7jQ9YB7klIz3RqVL+wNmeWI4=</ds:SignatureValue>
2761   <ds:KeyInfo>
2762     <ds:X509Data>
2763     <ds:X509Certificate>
2764     MIIICyJCCAJOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgakkCzAJBgNVBAYTAlVT
2765     MRIwEAYDVQQQIEw1XaXNjb25zaW4xEDAoBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
2766     FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLIEYJEAxZpc2lvbiBvZiBJ
2767     bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
2768     LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MV0xMDA2MDkwNDA3Mjc1MVowgYsx
2769     CzAJBgNVBAYTAlVTREwDwYDVQQQIEwNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2770     Ym9yMQ4wDAYDVQQKEwVWV0QFJRDEcMBoGAlUEAxMTc2hpYjEuaW50ZXJmZjZlMjEw
2771     dTEuMCUGCSqGSIB3DQEQJARYYcm9vdEBzaGlzMS5pbnRlcmluZDludWZWR1MIGfMA0G
2772     CSqGSIB3DQEQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhaXnXVIVTxx8vuRay+x50z7GJj
2773     IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIaOAPSZB113R6+KYIE7x4XAWIRCP+
2774     c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
2775     pmqOIFGTWQIDAQABox0wGzAMBGNVHRMBAf8EAJAAMASGA1UdDwQEAwIFoDANBgkq
2776     hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2777     qgi7lFV6MDkHmTvtqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIFz6QZAv2FU78pLX
2778     8I3bsbmRAUg4UP9hH6ABVq4KQKMKnxulxQxLhpRlylGPdiowMNTREg8cCx3w/w==
2779     </ds:X509Certificate>
2780     </ds:X509Data>
2781   </ds:KeyInfo>
2782 </ds:Signature></Assertion></Response>

```

2783

## 6 SAML Extensions

2784 The SAML schemas support extensibility. An example of an application that extends SAML assertions is  
2785 the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use  
2786 the extensibility features in SAML to create extension schemas.

2787 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML  
2788 elements can serve as the head element of a substitution group. However, SAML types are not defined  
2789 as *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only  
2790 elements and types that have been specifically designed to support extensibility.

### 6.1 Assertion Schema Extension

2792 The SAML assertion schema is designed to permit separate processing of the assertion package and the  
2793 statements it contains, if the extension mechanism is used for either part.

2794 The following elements are intended specifically for use as extension points in an extension schema;  
2795 their types are set to *abstract*, and are thus usable only as the base of a derived type:

2796 • <Condition>

2797 • <Statement>

2798 • The following elements that are directly usable as part of SAML MAY be extended:

2799 • <AuthenticationStatement>

2800 • <AuthorizationDecisionStatement>

2801 • <AttributeStatement>

2802 • <AudienceRestrictionCondition>

2803 The following elements are defined to allow elements from arbitrary namespaces within them, which  
2804 serves as a built-in extension point without requiring an extension schema:

2805 • <BaseIdentifier>

2806 • <SubjectConfirmationData>

2807 • <AttributeValue>

2808 • <Advice>

2809 • <AuthnContext>

### 6.2 Protocol Schema Extension

2811 The following SAML protocol elements are intended specifically for use as extension points in an  
2812 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived  
2813 type:

2814 • <Query>

2815 • <SubjectQuery>

2816 The following elements that are directly usable as part of SAML MAY be extended:

- 2817 • <Request>
- 2818 • <AuthenticationQuery>
- 2819 • <AuthorizationDecisionQuery>
- 2820 • <AttributeQuery>
- 2821 • <Response>

---

## 2822 7 SAML-Defined Identifiers

2823 The following sections define URI-based identifiers for common authentication methods, resource access  
2824 actions, and subject name identifier formats.

2825 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of  
2826 the most current RFC that specifies the protocol is used. URI references created specifically for SAML  
2827 have one of the following stems:

```
2828 urn:oasis:names:tc:SAML:1.0:  
2829 urn:oasis:names:tc:SAML:1.1:
```

### 2830 7.1 Authentication Method Identifiers

2831 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the  
2832 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although  
2833 both can refer to the same underlying mechanisms. An authentication statement with an  
2834 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The  
2835 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the  
2836 authentication statement does not provide the means to perform that authentication, such as a password,  
2837 key, or certificate.

2838 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,  
2839 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML  
2840 relying party to confirm that the request or message came from a system entity that corresponds to the  
2841 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method  
2842 that the relying party can use to do this in the future. This may or may not have any relationship to an  
2843 authentication that was performed previously. Unlike the authentication method, the subject confirmation  
2844 method may be accompanied by some piece of information, such as a certificate or key, that will allow  
2845 the relying party to perform the necessary check.

2846 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML  
2847 profiles specification [SAMLProf] for more information. Additional methods may be added by defining  
2848 new profiles or by private agreement.

2849 The following identifiers refer to SAML-specified authentication methods.

#### 2850 7.1.1 Password

2851 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

2852 The authentication was performed by means of a password.

#### 2853 7.1.2 Kerberos

2854 **URI:** urn:ietf:rfc:1510

2855 The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the  
2856 Needham-Schroeder symmetric key authentication mechanism [Needham78].

#### 2857 7.1.3 Secure Remote Password (SRP)

2858 **URI:** urn:ietf:rfc:2945

2859 The authentication was performed by means of Secure Remote Password protocol as specified in [RFC  
2860 2945].

#### 2861 **7.1.4 Hardware Token**

2862 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

2863 The authentication was performed using some (unspecified) hardware token.

#### 2864 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

2865 **URI:** urn:ietf:rfc:2246

2866 The authentication was performed using either the SSL or TLS protocol with certificate-based client  
2867 authentication. TLS is described in [RFC 2246].

#### 2868 **7.1.6 X.509 Public Key**

2869 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

2870 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2871 of an X.509 PKI [X.509][PKIX]. It may have been one of the mechanisms for which a more specific  
2872 identifier has been defined below.

#### 2873 **7.1.7 PGP Public Key**

2874 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

2875 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2876 of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more specific  
2877 identifier has been defined below.

#### 2878 **7.1.8 SPKI Public Key**

2879 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

2880 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2881 of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific identifier has  
2882 been defined below.

#### 2883 **7.1.9 XKMS Public Key**

2884 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

2885 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2886 of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more specific  
2887 identifier has been defined below.

#### 2888 **7.1.10 XML Digital Signature**

2889 **URI:** urn:ietf:rfc:3075

2890 The authentication was performed by means of an XML digital signature [RFC 3075].

2891 **7.1.11 Authentication Context**

2892 **URI:** urn:oasis:names:tc:SAML:2.0:am:authncontext

2893 The authentication method is described by the proximal <AuthnContext> element.

2894 **7.1.12 Unspecified**

2895 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

2896 The authentication was performed by an unspecified means.

2897 **7.2 Action Namespace Identifiers**

2898 The following identifiers MAY be used in the `Namespace` attribute of the <Action> element (see  
2899 Section Element <Action>) to refer to common sets of actions to perform on resources.

2900 **7.2.1 Read/Write/Execute/Delete/Control**

2901 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

2902 Defined actions:

2903 `Read Write Execute Delete Control`

2904 These actions are interpreted as follows:

2905 `Read`

2906 The subject may read the resource.

2907 `Write`

2908 The subject may modify the resource.

2909 `Execute`

2910 The subject may execute the resource.

2911 `Delete`

2912 The subject may delete the resource.

2913 `Control`

2914 The subject may specify the access control policy for the resource.

2915 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

2916 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

2917 Defined actions:

2918 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

2919 The actions specified in Section Read/Write/Execute/Delete/Control are interpreted in the same manner  
2920 described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively  
2921 specify that the stated permission is denied. Thus a subject described as being authorized to perform the  
2922 action `~Read` is affirmatively denied read permission.

2923 A SAML authority MUST NOT authorize both an action and its negated form.

### 2924 **7.2.3 Get/Head/Put/Post**

2925 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

2926 Defined actions:

2927 GET HEAD PUT POST

2928 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
2929 the GET action on a resource is authorized to retrieve it.

2930 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and  
2931 POST actions to the write permission. The correspondence is not exact however since an HTTP GET  
2932 operation may cause data to be modified and a POST operation may cause modification to a resource  
2933 other than the one specified in the request. For this reason a separate Action URI reference specifier is  
2934 provided.

### 2935 **7.2.4 UNIX File Permissions**

2936 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

2937 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
2938 notation.

2939 The action string is a four-digit numeric code:

2940 *extended user group world*

2941 Where the *extended* access permission has the value

2942 +2 if sgid is set

2943 +4 if suid is set

2944 The *user group* and *world* access permissions have the value

2945 +1 if execute permission is granted

2946 +2 if write permission is granted

2947 +4 if read permission is granted

2948 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read  
2949 and execute; and world read.

## 2950 **7.3 NameIdentifier Format Identifiers**

2951 The following identifiers MAY be used in the `Format` attribute of the `<NameIdentifier>` element (see  
2952 Section ) to refer to common formats for the content of the `<NameIdentifier>` element and the  
2953 associated processing rules, if any.

2954 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of  
2955 SAML.

### 2956 **7.3.1 Unspecified**

2957 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

2958 The interpretation of the content of the element is left to individual implementations.

### 2959 **7.3.2 Email Address**

2960 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

2961 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as  
2962 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that  
2963 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in  
2964 parentheses) after it, and is not surrounded by "<" and ">".

### 2965 **7.3.3 X.509 Subject Name**

2966 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

2967 Indicates that the content of the element is in the form specified for the contents of the  
2968 <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors  
2969 should note that the XML Signature specification specifies encoding rules for X.509 subject names that  
2970 differ from the rules given in IETF RFC 2253 [RFC 2253].

### 2971 **7.3.4 Windows Domain Qualified Name**

2972 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

2973 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
2974 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"  
2975 separator MAY be omitted.

### 2976 **7.3.5 Provider Identifier**

2977 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:provider

2978 Indicates that the content of the element is the identifier of a provider of SAML-based services (such as a  
2979 SAML authority) or a participant in SAML profiles (such as a service provider supporting the browser  
2980 profiles). Such an identifier can be used to make assertions about system entities that can issue SAML  
2981 requests, responses, and assertions.

### 2982 **7.3.6 Federated Identifier**

2983 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:federated

2984 Indicates that the content of the element is a persistent opaque identifier that corresponds to an identity  
2985 federation between an identity provider and a service provider (or affiliation of service providers).  
2986 Federated name identifiers generated by identity providers MUST be constructed using pseudo-random  
2987 values that have no discernible correspondence with the subject's actual identifier (for example,  
2988 username). The intent is to create a non-public pseudonym to prevent the discovery of the subject's  
2989 identity or activities. Federated name identifier values MUST NOT exceed a length of 256 characters.

2990 The element's content MUST contain the most recent identifier of the subject set by the identity provider.

2991 The element's `NameQualifier` attribute, if present, MUST contain the name of the identity provider  
2992 participating in the identity federation. It MAY be omitted if the value can be derived from the context of  
2993 the message containing the element, such as the issuer of an assertion.

2994 The element's `SPNameQualifier` attribute, if present, MUST contain the name of the service provider  
2995 or affiliation of providers participating in the identity federation. It MAY be omitted if the element is  
2996 contained in a message intended only for consumption directly by the service provider, and the value  
2997 would be the name of that service provider.

2998 The element's `SPProvidedIdentifier` attribute MUST contain the alternative identifier of the subject  
2999 most recently set by the service provider or affiliation, if any. If no such identifier has been established,  
3000 than the attribute MUST be omitted.

3001 Federated identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear  
3002 text with providers other than the providers that have established the identity federation. Furthermore,  
3003 they MUST NOT appear in log files or similar locations without appropriate controls and protections.  
3004 Deployments without such requirements are free to use other kinds of identifiers in their SAML  
3005 exchanges.

3006 Note also that while federated identifiers are typically used to reflect an account linking relationship  
3007 between a pair of providers, a service provider is not obligated to recognize or make use of the long term  
3008 nature of the persistent identifier or establish such a link. Such a "one-sided" identity federation is not  
3009 discernibly different and does not affect the behavior of the identity provider or any processing rules  
3010 specific to federated identifiers in the protocols defined in this specification.

### 3011 **7.3.7 Transient Identifier**

3012 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

3013 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated  
3014 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated  
3015 in accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of  
3016 256 characters.

3017 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier  
3018 represents a transient and temporary identity federation, as described in Section Federated Identifier. In  
3019 such a case, they MAY be omitted in accordance with the rules specified in that section.

## 3020 **7.4 Attribute NameFormat Identifiers**

3021 The following identifiers MAY be used in the `NameFormat` attribute defined on the  
3022 **AttributeDesignatorType** complex type (see Section x) to refer to the classification of the attribute name  
3023 for purposes of interpreting the name.

### 3024 **7.4.1 Unspecified**

3025 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:unspecified

3026 The interpretation of the attribute name is left to individual implementations.

### 3027 **7.4.2 URI Reference**

3028 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:uri

3029 The attribute name follows the convention for URI references [RFC 2396], for example as used in  
3030 XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is  
3031 application-specific. [See the Baseline Identities and Attributes specification \[SAMLBaseAtts\] for a full](#)  
3032 [discussion of representing names of XACML, X.500, and LDAP attributes.](#)

## 3033 7.5 Attribute ValueType Identifiers

3034 The following identifier MAY be used in the `ValueType` attribute defined on the  
3035 **AttributeDesignatorType** complex type (see Section x) to refer to the URI-based datatype of the  
3036 desired or supplied attribute.

### 3037 7.5.1 **Application-SpecificUnspecified Value Type**

3038 **URI:** urn:oasis:names:tc:SAML:2.0:valuetype-format:[unspecifiedappSpecific](#)

3039 Indicates that the datatype of the desired or supplied attribute is application-specific. Note that any  
3040 `ValueType` setting (default or explicit) in an attribute query, including this setting, needs to be exactly  
3041 matched (in addition to other exact matches) in order for an attribute to be returned.

3042

---

## 8 References

3043

3044 The following works are cited in the body of this specification.

### 8.1 Normative References

- 3046 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web  
3047 Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3048 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
3049 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.  
3050 Note that this specification normatively references [Schema2], listed below.
- 3051 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
3052 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3053 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World  
3054 Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3055 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing,  
3056 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web  
3057 Consortium. Note that this specification normatively references [XMLEnc-XSD],  
3058 listed below.
- 3059 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium.  
3060 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3061 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January  
3062 1999. <http://www.w3.org/TR/REC-xml-names>.
- 3063 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web  
3064 Consortium, February 2002. <http://www.w3.org/TR/xmldsig-core/>. Note that this  
3065 specification normatively references [XMLSig-XSD], listed below.
- 3066 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium.  
3067 [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)  
3068 [schema.xsd](http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd).

### 8.2 Non-Normative References

- 3069 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty  
3070 Alliance Project, January 2003,  
3071 [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)  
3072 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3074 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of*  
3075 *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December  
3076 1978.
- 3077 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann. *PGP Message Exchange Formats*.  
3078 IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 3079 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*  
3080 *Certificate and CRL Profile*. IETF RFC 2459, January 1999.  
3081 <http://www.ietf.org/rfc/rfc2459.txt>.
- 3082 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.  
3083 IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3084 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF  
3085 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

3086 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January  
3087 1999. <http://www.ietf.org/rfc/rfc2246.txt>.

3088 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*  
3089 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.  
3090 <http://www.ietf.org/rfc/rfc2253.txt>.

3091 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF  
3092 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3093 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.  
3094 <http://www.ietf.org/rfc/rfc2630.txt>.

3095 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.  
3096 <http://www.ietf.org/rfc/rfc2822.txt>.

3097 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,  
3098 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3099 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF  
3100 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3101 [SAMLAuthnCxt] J. Kemp. Authentication Context for the -OASIS Security Assertion Markup  
3102 Language (SAML). OASIS, February 2004. Document ID sstc-saml-authn-  
3103 context-2.0. <http://www.oasis-open.org/committees/security/>.

3104 [SAMLBaseAtts] J. Hughes and P. Mishra. Baseline Identities and Attributes for the OASIS  
3105 Security Assertion Markup Language (SAML). OASIS, March 2004. Document ID  
3106 sstc-saml-baseline-atts-2.0. <http://www.oasis-open.org/committees/security/>.

3107 [SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*  
3108 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0.  
3109 <http://www.oasis-open.org/committees/security/>.

3110 [SAMLProf] E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*  
3111 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0.  
3112 <http://www.oasis-open.org/committees/security/>.

3113 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*  
3114 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
3115 oasis-sstc-saml-conform-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3116 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3117 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*  
3118 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf)  
3119 [open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3120 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*  
3121 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1.  
3122 HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3123 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
[open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3124 [SAMLPSchema] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID  
3125 oasis-sstc-saml-schema-protocol-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3126 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3127 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*  
3128 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
3129 oasis-sstc-saml-sec-consider-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3130 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3131 [SAMLXSD] E. Maler et al. *SAML assertion schema*. OASIS, September 2003. Document ID  
3132 oasis-sstc-saml-schema-assertion-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3133 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3134	<b>[SPKI]</b>	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> . IETF RFC 2693, September 1999. <a href="http://www.ietf.org/rfc/rfc2693.txt">http://www.ietf.org/rfc/rfc2693.txt</a> .
3135		
3136		
3137	<b>[UNICODE-C]</b>	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. <a href="http://www.unicode.org/unicode/reports/tr15/tr15-21.html">http://www.unicode.org/unicode/reports/tr15/tr15-21.html</a> .
3138		
3139	<b>[W3C-CHAR]</b>	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a> .
3140		
3141	<b>[W3C-CharMod]</b>	M. J. Dürst. <i>Character Model for the World Wide Web 1.0</i> . World Wide Web Consortium, April, 2002. <a href="http://www.w3.org/TR/charmod/">http://www.w3.org/TR/charmod/</a> .
3142		
3143	<b>[X.500]</b>	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
3144		
3145	<b>[XACML]</b>	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml">http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml</a> .
3146		
3147		
3148	<b>[XKMS]</b>	W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp. XML Key Management Specification (XKMS). W3C Note 30 March 2001. <a href="http://www.w3.org/TR/xkms/">http://www.w3.org/TR/xkms/</a> .
3149		
3150		

---

3151 **Appendix A. Acknowledgments**

3152 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
3153 Committee, whose voting members at the time of publication were:

- 3154 • @@

## 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.
<a href="http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf">http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf</a>			
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).
<a href="http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf">http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf</a>			
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.
<a href="http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf">http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf</a> <a href="http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf">http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf</a>			
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 <b>W-2</b> , 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.
<a href="http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf">http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf</a>			
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol ( <b>W-2</b> ), 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 ( <b>W-1</b> ); still unfinished.
<a href="http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf">http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf</a>			
06	20 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added AssertionURIReference ( <b>W-19</b> ), a proposal for ProxyRestrictionCondition, and a proposal for AuthNRequest/Response (related to many work items). Fleshed out LogoutRequest/Response ( <b>W-1</b> ). Implemented the freezing of authZ decision statement functionality ( <b>W-28b</b> ).
<a href="http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf">http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf</a>			

Rev	Date	By Whom	What
07	7 Mar 2004	Scott Cantor, Eve Maler	<p>Implemented new arrangement for subject information and decision on KeyInfo description, as agreed at 2 Mar 2004 telecon.</p> <p>Adjusted normative language around subject "matching" rules based on subject changes.</p> <p>Revised AuthnRequest proposal based on those changes and feedback from list and focus calls.</p> <p>Incorporated additional schema and processing rules related to ECP and proxying use cases from ID-FF.</p> <p>Added AuthnContext to AuthenticationStatement.</p> <p>Added NameIdentifierMapping protocol (<b>W-2</b>).</p>
<a href="http://www.oasis-open.org/committees/download.php/5790/sstc-saml-core-2.0-draft-07-diff.pdf">http://www.oasis-open.org/committees/download.php/5790/sstc-saml-core-2.0-draft-07-diff.pdf</a>			
08	15 Mar 2004	Scott Cantor, Eve Maler	<p>Added ArtifactRequest/Response pair as a new protocol.</p> <p>Implemented proposed W-28a attribute changes (rev 03 of the proposal, reflecting focus group input).</p>
<a href="http://www.oasis-open.org/committees/download.php/5951/sstc-saml-core-2.0-draft-08-diff.pdf">http://www.oasis-open.org/committees/download.php/5951/sstc-saml-core-2.0-draft-08-diff.pdf</a>			
09	8 Apr 2004	Eve Maler	<p>Minor cleanup, plus decisions from March-April 2004 F2F meeting: Moved Signature element up in Assertion contents. Clarified that DoNotCacheCondition has one-time-use semantics. Made NameFormat on the Attribute element clearly optional. Changed the default ValueType identifier name. Added the ability to put arbitrary attributes on the AttributeDesignator element. Removed Source on the Attribute element. Changed the content of Extensions in the Request element to ##other. Removed the restriction saying only federated identifiers could be replaced and set with the termination protocol. Changed Reason on the LogoutRequest element to be a URI reference. Made SessionIndex in the LogoutRequest element globally declared. Added bibliographic references to the new SAML specs.</p>

3156

3157

---

## Appendix C. Notices

3158 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
3159 might be claimed to pertain to the implementation or use of the technology described in this document or  
3160 the extent to which any license under such rights might or might not be available; neither does it  
3161 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
3162 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
3163 made available for publication and any assurances of licenses to be made available, or the result of an  
3164 attempt made to obtain a general license or permission for the use of such proprietary rights by  
3165 implementors or users of this specification, can be obtained from the OASIS Executive Director.

3166 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
3167 or other proprietary rights which may cover technology that may be required to implement this  
3168 specification. Please address the information to the OASIS Executive Director.

3169 **Copyright © OASIS Open 2004. All Rights Reserved.**

3170 This document and translations of it may be copied and furnished to others, and derivative works that  
3171 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
3172 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright  
3173 notice and this paragraph are included on all such copies and derivative works. However, this document  
3174 itself may not be modified in any way, such as by removing the copyright notice or references to OASIS,  
3175 except as needed for the purpose of developing OASIS specifications, in which case the procedures for  
3176 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required  
3177 to translate it into languages other than English.

3178 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
3179 or assigns.

3180 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
3181 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
3182 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS  
3183 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR  
3184 PURPOSE.