



# Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0

OASIS Standard, 15 March 2005

**Document identifier:**

saml-profiles-2.0-os

**Location:**

<http://docs.oasis-open.org/security/saml/v2.0/>

**Editors:**

John Hughes, Atos Origin  
Scott Cantor, Internet2  
Jeff Hodges, Neustar  
Frederick Hirsch, Nokia  
Prateek Mishra, Principal Identity  
Rob Philpott, RSA Security  
Eve Maler, Sun Microsystems

**SAML V2.0 Contributors:**

Conor P. Cahill, AOL  
John Hughes, Atos Origin  
Hal Lockhart, BEA Systems  
Michael Beach, Boeing  
Rebekah Metz, Booz Allen Hamilton  
Rick Randall, Booz Allen Hamilton  
Thomas Wisniewski, Entrust  
Irving Reid, Hewlett-Packard  
Paula Austel, IBM  
Maryann Hondo, IBM  
Michael McIntosh, IBM  
Tony Nadalin, IBM  
Nick Ragouzis, Individual  
Scott Cantor, Internet2  
RL 'Bob' Morgan, Internet2  
Peter C Davis, Neustar  
Jeff Hodges, Neustar  
Frederick Hirsch, Nokia  
John Kemp, Nokia  
Paul Madsen, NTT  
Steve Anderson, OpenNetwork  
Prateek Mishra, Principal Identity  
John Linn, RSA Security  
Rob Philpott, RSA Security  
Jahan Moreh, Sigaba  
Anne Anderson, Sun Microsystems

45 Eve Maler, Sun Microsystems  
46 Ron Monzillo, Sun Microsystems  
47 Greg Whitehead, Trustgenix

48 **Abstract:**

49 This specification defines profiles for the use of SAML assertions and request-response  
50 messages in communications protocols and frameworks, as well as profiles for SAML attribute  
51 value syntax and naming conventions.

52 **Status:**

53 This is an **OASIS Standard** document produced by the Security Services Technical Committee. It  
54 was approved by the OASIS membership on 1 March 2005.

55 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)  
56 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by filling out the web form located  
57 at [http://www.oasis-open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security). The  
58 committee will publish on its web page (<http://www.oasis-open.org/committees/security>) a catalog  
59 of any changes made to this document.

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

## Table of Contents

65	1 Introduction.....	7
66	1.1 Profile Concepts.....	7
67	1.2 Notation.....	7
68	2 Specification of Additional Profiles.....	10
69	2.1 Guidelines for Specifying Profiles.....	10
70	2.2 Guidelines for Specifying Attribute Profiles.....	10
71	3 Confirmation Method Identifiers.....	12
72	3.1 Holder of Key.....	12
73	3.2 Sender Vouches.....	12
74	3.3 Bearer.....	13
75	4 SSO Profiles of SAML.....	14
76	4.1 Web Browser SSO Profile.....	14
77	4.1.1 Required Information.....	14
78	4.1.2 Profile Overview.....	14
79	4.1.3 Profile Description.....	16
80	4.1.3.1 HTTP Request to Service Provider.....	16
81	4.1.3.2 Service Provider Determines Identity Provider.....	16
82	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	16
83	4.1.3.4 Identity Provider Identifies Principal.....	17
84	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	17
85	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	17
86	4.1.4 Use of Authentication Request Protocol.....	18
87	4.1.4.1 <AuthnRequest> Usage.....	18
88	4.1.4.2 <Response> Usage.....	18
89	4.1.4.3 <Response> Message Processing Rules.....	19
90	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	20
91	4.1.4.5 POST-Specific Processing Rules.....	20
92	4.1.5 Unsolicited Responses.....	20
93	4.1.6 Use of Metadata.....	20
94	4.2 Enhanced Client or Proxy (ECP) Profile.....	21
95	4.2.1 Required Information.....	21
96	4.2.2 Profile Overview.....	21
97	4.2.3 Profile Description.....	24
98	4.2.3.1 ECP issues HTTP Request to Service Provider.....	24
99	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	25
100	4.2.3.3 ECP Determines Identity Provider.....	25
101	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	25
102	4.2.3.5 Identity Provider Identifies Principal.....	25
103	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	26
104	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	26
105	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	26
106	4.2.4 ECP Profile Schema Usage.....	26
107	4.2.4.1 PAOS Request Header Block: SP to ECP.....	27
108	4.2.4.2 ECP Request Header Block: SP to ECP.....	28
109	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	28

110	4.2.4.4 ECP Response Header Block: IdP to ECP.....	29
111	4.2.4.5 PAOS Response Header Block: ECP to SP.....	30
112	4.2.5 Security Considerations.....	31
113	4.3 Identity Provider Discovery Profile.....	31
114	4.3.1 Common Domain Cookie.....	31
115	4.3.2 Setting the Common Domain Cookie.....	32
116	4.3.3 Obtaining the Common Domain Cookie.....	32
117	4.4 Single Logout Profile.....	32
118	4.4.1 Required Information.....	33
119	4.4.2 Profile Overview.....	33
120	4.4.3 Profile Description.....	34
121	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	34
122	4.4.3.2 Identity Provider Determines Session Participants.....	35
123	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	35
124	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	36
125	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	36
126	4.4.4 Use of Single Logout Protocol.....	37
127	4.4.4.1 <LogoutRequest> Usage.....	37
128	4.4.4.2 <LogoutResponse> Usage.....	37
129	4.4.5 Use of Metadata.....	37
130	4.5 Name Identifier Management Profile.....	37
131	4.5.1 Required Information.....	38
132	4.5.2 Profile Overview.....	38
133	4.5.3 Profile Description.....	38
134	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	39
135	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	39
136	4.5.4 Use of Name Identifier Management Protocol.....	40
137	4.5.4.1 <ManageNameIDRequest> Usage.....	40
138	4.5.4.2 <ManageNameIDResponse> Usage.....	40
139	4.5.5 Use of Metadata.....	40
140	5 Artifact Resolution Profile.....	41
141	5.1 Required Information.....	41
142	5.2 Profile Overview.....	41
143	5.3 Profile Description.....	42
144	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	42
145	5.3.2 <ArtifactResponse> issued by Responding Entity.....	42
146	5.4 Use of Artifact Resolution Protocol.....	42
147	5.4.1 <ArtifactResolve> Usage.....	42
148	5.4.2 <ArtifactResponse> Usage.....	43
149	5.5 Use of Metadata.....	43
150	6 Assertion Query/Request Profile.....	44
151	6.1 Required Information.....	44
152	6.2 Profile Overview.....	44
153	6.3 Profile Description.....	45
154	6.3.1 Query/Request issued by SAML Requester.....	45
155	6.3.2 <Response> issued by SAML Authority.....	45

156	6.4 Use of Query/Request Protocol.....	45
157	6.4.1 Query/Request Usage.....	45
158	6.4.2 <Response> Usage.....	45
159	6.5 Use of Metadata.....	46
160	7 Name Identifier Mapping Profile.....	47
161	7.1 Required Information.....	47
162	7.2 Profile Overview.....	47
163	7.3 Profile Description.....	48
164	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	48
165	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	48
166	7.4 Use of Name Identifier Mapping Protocol.....	48
167	7.4.1 <NameIDMappingRequest> Usage.....	48
168	7.4.2 <NameIDMappingResponse> Usage.....	48
169	7.4.2.1 Limiting Use of Mapped Identifier.....	49
170	7.5 Use of Metadata.....	49
171	8 SAML Attribute Profiles.....	50
172	8.1 Basic Attribute Profile.....	50
173	8.1.1 Required Information.....	50
174	8.1.2 SAML Attribute Naming.....	50
175	8.1.2.1 Attribute Name Comparison.....	50
176	8.1.3 Profile-Specific XML Attributes.....	50
177	8.1.4 SAML Attribute Values.....	50
178	8.1.5 Example.....	50
179	8.2 X.500/LDAP Attribute Profile.....	50
180	8.2.1 Required Information.....	51
181	8.2.2 SAML Attribute Naming.....	51
182	8.2.2.1 Attribute Name Comparison.....	51
183	8.2.3 Profile-Specific XML Attributes.....	51
184	8.2.4 SAML Attribute Values.....	51
185	8.2.5 Profile-Specific Schema.....	52
186	8.2.6 Example.....	53
187	8.3 UUID Attribute Profile.....	53
188	8.3.1 Required Information.....	53
189	8.3.2 UUID and GUID Background.....	53
190	8.3.3 SAML Attribute Naming.....	54
191	8.3.3.1 Attribute Name Comparison.....	54
192	8.3.4 Profile-Specific XML Attributes.....	54
193	8.3.5 SAML Attribute Values.....	54
194	8.3.6 Example.....	54
195	8.4 DCE PAC Attribute Profile.....	55
196	8.4.1 Required Information.....	55
197	8.4.2 PAC Description.....	55
198	8.4.3 SAML Attribute Naming.....	55
199	8.4.3.1 Attribute Name Comparison.....	55

200	8.4.4 Profile-Specific XML Attributes.....	55
201	8.4.5 SAML Attribute Values.....	56
202	8.4.6 Attribute Definitions.....	56
203	8.4.6.1 Realm.....	56
204	8.4.6.2 Principal.....	57
205	8.4.6.3 Primary Group.....	57
206	8.4.6.4 Groups.....	57
207	8.4.6.5 Foreign Groups.....	57
208	8.4.7 Example.....	58
209	8.5 XACML Attribute Profile.....	58
210	8.5.1 Required Information.....	59
211	8.5.2 SAML Attribute Naming.....	59
212	8.5.2.1 Attribute Name Comparison.....	59
213	8.5.3 Profile-Specific XML Attributes.....	59
214	8.5.4 SAML Attribute Values.....	59
215	8.5.5 Profile-Specific Schema.....	59
216	8.5.6 Example.....	60
217	9 References.....	61
218	Appendix A. Acknowledgments.....	64
219	Appendix B. Notices.....	66

---

## 220 1 Introduction

221 This document specifies profiles that define the use of SAML assertions and request-response messages  
222 in communications protocols and frameworks, as well as profiles that define SAML attribute value syntax  
223 and naming conventions.

224 The SAML assertions and protocols specification [SAMLCore] defines the SAML assertions and request-  
225 response protocol messages themselves, and the SAML bindings specification [SAMLBind] defines  
226 bindings of SAML protocol messages to underlying communications and messaging protocols. The SAML  
227 conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

### 228 1.1 Profile Concepts

229 One type of SAML profile outlines a set of rules describing how to embed SAML assertions into and  
230 extract them from a framework or protocol. Such a profile describes how SAML assertions are embedded  
231 in or combined with other objects (for example, files of various types, or protocol data units of  
232 communication protocols) by an originating party, communicated from the originating party to a receiving  
233 party, and subsequently processed at the destination. A particular set of rules for embedding SAML  
234 assertions into and extracting them from a specific class of <FOO> objects is termed a <FOO> *profile of*  
235 *SAML*.

236 For example, a SOAP profile of SAML describes how SAML assertions can be added to SOAP messages,  
237 how SOAP headers are affected by SAML assertions, and how SAML-related error states should be  
238 reflected in SOAP messages.

239 Another type of SAML profile defines a set of constraints on the use of a general SAML protocol or  
240 assertion capability for a particular environment or context of use. Profiles of this nature may constrain  
241 optionality, require the use of specific SAML functionality (for example, attributes, conditions, or bindings),  
242 and in other respects define the processing rules to be followed by profile actors.

243 A particular example of the latter are those that address SAML attributes. The SAML <Attribute>  
244 element provides a great deal of flexibility in attribute naming, value syntax, and including in-band  
245 metadata through the use of XML attributes. Interoperability is achieved by constraining this flexibility  
246 when warranted by adhering to profiles that define how to use these elements with greater specificity than  
247 the generic rules defined by [SAMLCore].

248 Attribute profiles provide the definitions necessary to constrain SAML attribute expression when dealing  
249 with particular types of attribute information or when interacting with external systems or other open  
250 standards that require greater strictness.

251 The intent of this specification is to specify a selected set of profiles of various kinds in sufficient detail to  
252 ensure that independently implemented products will interoperate.

253 For other terms and concepts that are specific to SAML, refer to the SAML glossary [SAMLGloss].

### 254 1.2 Notation

255 This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative  
256 text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In  
257 cases of disagreement between the SAML profile schema documents and schema listings in this  
258 specification, the schema documents take precedence. Note that in some cases the normative text of this  
259 specification imposes constraints beyond those indicated by the schema documents.

260 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD  
261 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as

262 described in IETF RFC 2119 [RFC2119].

263 Listings of productions or other normative code appear like this.

264 Example code listings appear like this.

265 **Note:** Notes like this are sometimes used to highlight non-normative commentary.

266 Conventional XML namespace prefixes are used throughout this specification to stand for their respective  
267 namespaces as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
md:	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace [SAMLMeta].
ecp:	urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp	This is the SAML V2.0 ECP profile namespace, specified in this document and in a schema [SAMLECP-xsd].
ds:	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
xenc:	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
SOAP-ENV:	http://schemas.xmlsoap.org/soap/envelope	This is the SOAP V1.1 namespace [SOAP1.1].
paos:	urn:liberty:paos:2003-08	This is the Liberty Alliance PAOS namespace.
dce:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE	This is the SAML V2.0 DCE PAC attribute profile namespace, specified in this document and in a schema [SAMLDCExsd].
x500:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500	This is the SAML V2.0 X.500/LDAP attribute profile namespace, specified in this document and in a schema [SAMLX500-xsd].
xacmlprof:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML	This is the SAML V2.0 XACML attribute profile namespace, specified in this document and in a schema [SAMLXAC-xsd].
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

268 This specification uses the following typographical conventions in text: <SAMLElement>,  
269 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword. In some cases, angle brackets  
270 are used to indicate non-terminals, rather than XML elements; the intent will be clear from the context.

271

## 2 Specification of Additional Profiles

272 This specification defines a selected set of profiles, but others will possibly be developed in the future. It is  
273 not possible for the OASIS Security Services Technical Committee to standardize all of these additional  
274 profiles for two reasons: it has limited resources and it does not own the standardization process for all of  
275 the technologies used. The following sections offer guidelines for specifying profiles.

276 The SSTC welcomes proposals for new profiles. OASIS members may wish to submit these proposals for  
277 consideration by the SSTC in a future version of this specification. Other members may simply wish to  
278 inform the committee of their work related to SAML. Please refer to the SSTC website [SAMLWeb] for  
279 further details on how to submit such proposals to the SSTC.

### 2.1 Guidelines for Specifying Profiles

281 This section provides a checklist of issues that MUST be addressed by each profile.

- 282 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the  
283 author, and provide reference to previously defined profiles that the new profile updates or  
284 obsoletes.
- 285 2. Describe the set of interactions between parties involved in the profile. Any restrictions on  
286 applications used by each party and the protocols involved in each interaction must be explicitly  
287 called out.
- 288 3. Identify the parties involved in each interaction, including how many parties are involved and  
289 whether intermediaries may be involved.
- 290 4. Specify the method of authentication of parties involved in each interaction, including whether  
291 authentication is required and acceptable authentication types.
- 292 5. Identify the level of support for message integrity, including the mechanisms used to ensure  
293 message integrity.
- 294 6. Identify the level of support for confidentiality, including whether a third party may view the contents  
295 of SAML messages and assertions, whether the profile requires confidentiality, and the  
296 mechanisms recommended for achieving confidentiality.
- 297 7. Identify the error states, including the error states at each participant, especially those that receive  
298 and process SAML assertions or messages.
- 299 8. Identify security considerations, including analysis of threats and description of countermeasures.
- 300 9. Identify SAML confirmation method identifiers defined and/or utilized by the profile.
- 301 10. Identify relevant SAML metadata defined and/or utilized by the profile.

### 2.2 Guidelines for Specifying Attribute Profiles

302 This section provides a checklist of items that MUST in particular be addressed by attribute profiles.

- 304 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the  
305 author, and provide reference to previously defined profiles that the new profile updates or  
306 obsoletes.
- 307 2. Syntax and restrictions on the acceptable values of the `NameFormat` and `Name` attributes of SAML  
308 `<Attribute>` elements.
- 309 3. Any additional namespace-qualified XML attributes defined by the profile that may be used in SAML  
310 `<Attribute>` elements.

- 311 4. Rules for determining the equality of SAML `<Attribute>` elements as defined by the profile, for  
312 use when processing attributes, queries, etc.
- 313 5. Syntax and restrictions on values acceptable in the SAML `<AttributeValue>` element, including  
314 whether the `xsi:type` XML attribute can or should be used.

---

## 3 Confirmation Method Identifiers

315

316 The SAML assertion and protocol specification [SAMLCore] defines the `<SubjectConfirmation>`  
317 element as a `Method` plus optional `<SubjectConfirmationData>`. The `<SubjectConfirmation>`  
318 element SHOULD be used by the relying party to confirm that the request or message came from a  
319 system entity that is associated with the subject of the assertion, within the context of a particular profile.

320 The `Method` attribute indicates the specific method that the relying party should use to make this  
321 determination. This may or may not have any relationship to an authentication that was performed  
322 previously. Unlike the authentication context, the subject confirmation method will often be accompanied  
323 by additional information, such as a certificate or key, in the `<SubjectConfirmationData>` element  
324 that will allow the relying party to perform the necessary verification. A common set of attributes is also  
325 defined and MAY be used to constrain the conditions under which the verification can take place.

326 It is anticipated that profiles will define and use several different values for `<ConfirmationMethod>`,  
327 each corresponding to a different SAML usage scenario. The following methods are defined for use by  
328 profiles defined within this specification and other profiles that find them useful.

### 3.1 Holder of Key

329

330 **URI:** urn:oasis:names:tc:SAML:2.0:cm:holder-of-key

331 One or more `<ds:KeyInfo>` elements MUST be present within the `<SubjectConfirmationData>`  
332 element. An `xsi:type` attribute MAY be present in the `<SubjectConfirmationData>` element and, if  
333 present, MUST be set to **saml:KeyInfoConfirmationDataType** (the namespace prefix is arbitrary but  
334 must reference the SAML assertion namespace).

335 As described in [XMLSig], each `<ds:KeyInfo>` element holds a key or information that enables an  
336 application to obtain a key. The holder of a specified key is considered to be the subject of the assertion  
337 by the asserting party.

338 Note that in accordance with [XMLSig], each `<ds:KeyInfo>` element MUST identify a single  
339 cryptographic key. Multiple keys MAY be identified with separate `<ds:KeyInfo>` elements, such as when  
340 different confirmation keys are needed for different relying parties.

341 **Example:** The holder of the key named "By-Tor" or the holder of the key named "Snow Dog" can confirm  
342 itself as the subject.

```
343 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">  
344   <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
345     <ds:KeyInfo>  
346       <ds:KeyName>By-Tor</ds:KeyName>  
347     </ds:KeyInfo>  
348     <ds:KeyInfo>  
349       <ds:KeyName>Snow Dog</ds:KeyName>  
350     </ds:KeyInfo>  
351   </SubjectConfirmationData>  
352 </SubjectConfirmation>
```

### 3.2 Sender Vouches

353

354 **URI:** urn:oasis:names:tc:SAML:2.0:cm:sender-vouches

355 Indicates that no other information is available about the context of use of the assertion. The relying party  
356 SHOULD utilize other means to determine if it should process the assertion further, subject to optional  
357 constraints on confirmation using the attributes that MAY be present in the  
358 `<SubjectConfirmationData>` element, as defined by [SAMLCore].

359 **3.3 Bearer**

360 **URI:** urn:oasis:names:tc:SAML:2.0:cm:bearer

361 The subject of the assertion is the bearer of the assertion, subject to optional constraints on confirmation  
362 using the attributes that MAY be present in the <SubjectConfirmationData> element, as defined by  
363 [SAMLCore].

364 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered  
365 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March  
366 19<sup>th</sup>, 2004, in response to a request with ID "\_1234567890".

```
367 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
368   <SubjectConfirmationData InResponseTo="_1234567890"  
369     Recipient="https://www.serviceprovider.com/saml/consumer"  
370     NotOnOrAfter="2004-03-19T13:27:00Z"  
371   </SubjectConfirmationData>  
372 </SubjectConfirmation>
```

---

## 373 4 SSO Profiles of SAML

374 A set of profiles is defined to support single sign-on (SSO) of browsers and other client devices.

- 375 • A web browser-based profile of the Authentication Request protocol in [SAMLCore] is defined to  
376 support web single sign-on, supporting Scenario 1-1 of the original SAML requirements document .
- 377 • An additional web SSO profile is defined to support enhanced clients.
- 378 • A profile of the Single Logout and Name Identifier Management protocols in [SAMLCore] is defined  
379 over both front-channel (browser) and back-channel bindings.
- 380 • An additional profile is defined for identity provider discovery using cookies.

### 381 4.1 Web Browser SSO Profile

382 In the scenario supported by the web browser SSO profile, a web user either accesses a resource at a  
383 service provider, or accesses an identity provider such that the service provider and desired resource are  
384 understood or implicit. The web user authenticates (or has already authenticated) to the identity provider,  
385 which then produces an authentication assertion (possibly with input from the service provider) and the  
386 service provider consumes the assertion to establish a security context for the web user. During this  
387 process, a name identifier might also be established between the providers for the principal, subject to the  
388 parameters of the interaction and the consent of the parties.

389 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction  
390 with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.

391 It is assumed that the user is using a standard commercial browser and can authenticate to the identity  
392 provider by some means outside the scope of SAML.

#### 393 4.1.1 Required Information

394 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser

395 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

396 **SAML Confirmation Method Identifiers:** The SAML V2.0 "bearer" confirmation method identifier,  
397 urn:oasis:names:tc:SAML:2.0:cm:bearer, is used by this profile.

398 **Description:** Given below.

399 **Updates:** SAML V1.1 browser artifact and POST profiles and bearer confirmation method.

#### 400 4.1.2 Profile Overview

401 Figure 1 illustrates the basic template for achieving SSO. The following steps are described by the profile.  
402 Within an individual step, there may be one or more actual message exchanges depending on the binding  
403 used for that step and other implementation-dependent behavior.

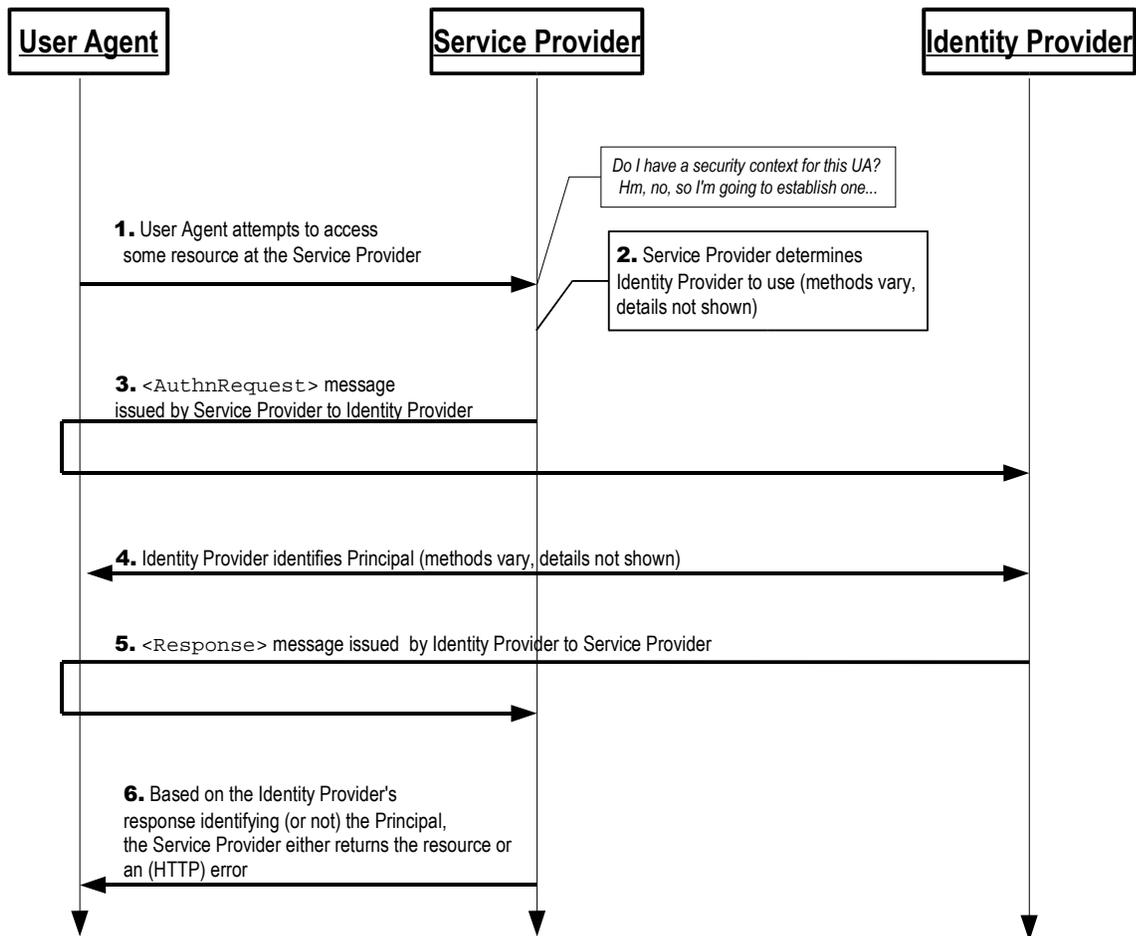


Figure 1

404 **1. HTTP Request to Service Provider**

405 In step 1, the principal, via an HTTP User Agent, makes an HTTP request for a secured resource  
406 at the service provider without a security context.

407 **2. Service Provider Determines Identity Provider**

408 In step 2, the service provider obtains the location of an endpoint at an identity provider for the  
409 authentication request protocol that supports its preferred binding. The means by which this is  
410 accomplished is implementation-dependent. The service provider MAY use the SAML identity  
411 provider discovery profile described in Section 4.3.

412 **3. <AuthnRequest> issued by Service Provider to Identity Provider**

413 In step 3, the service provider issues an <AuthnRequest> message to be delivered by the user  
414 agent to the identity provider. Either the HTTP Redirect, HTTP POST, or HTTP Artifact binding  
415 can be used to transfer the message to the identity provider through the user agent.

416 **4. Identity Provider identifies Principal**

417 In step 4, the principal is identified by the identity provider by some means outside the scope of  
418 this profile. This may require a new act of authentication, or it may reuse an existing authenticated  
419 session.

## 420 5. Identity Provider issues <Response> to Service Provider

421 In step 5, the identity provider issues a <Response> message to be delivered by the user agent  
422 to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer  
423 the message to the service provider through the user agent. The message may indicate an error,  
424 or will include (at least) an authentication assertion. The HTTP Redirect binding MUST NOT be  
425 used, as the response will typically exceed the URL length permitted by most user agents.

## 426 6. Service Provider grants or denies access to Principal

427 In step 6, having received the response from the identity provider, the service provider can  
428 respond to the principal's user agent with its own error, or can establish its own security context  
429 for the principal and return the requested resource.

430 Note that an identity provider can initiate this profile at step 5 and issue a <Response> message to a  
431 service provider without the preceding steps.

## 432 4.1.3 Profile Description

433 If the profile is initiated by the service provider, start with Section 4.1.3.1. If initiated by the identity  
434 provider, start with Section 4.1.3.5. In the descriptions below, the following are referred to:

### 435 Single Sign-On Service

436 This is the authentication request protocol endpoint at the identity provider to which the  
437 <AuthnRequest> message (or artifact representing it) is delivered by the user agent.

### 438 Assertion Consumer Service

439 This is the authentication request protocol endpoint at the service provider to which the  
440 <Response> message (or artifact representing it) is delivered by the user agent.

### 441 4.1.3.1 HTTP Request to Service Provider

442 If the first access is to the service provider, an arbitrary request for a resource can initiate the profile.  
443 There are no restrictions on the form of the request. The service provider is free to use any means it  
444 wishes to associate the subsequent interactions with the original request. Each of the bindings provide a  
445 RelayState mechanism that the service provider MAY use to associate the profile exchange with the  
446 original request. The service provider SHOULD reveal as little of the request as possible in the RelayState  
447 value unless the use of the profile does not require such privacy measures.

### 448 4.1.3.2 Service Provider Determines Identity Provider

449 This step is implementation-dependent. The service provider MAY use the SAML identity provider  
450 discovery profile, described in Section 4.3. The service provider MAY also choose to redirect the user  
451 agent to another service that is able to determine an appropriate identity provider. In such a case, the  
452 service provider may issue an <AuthnRequest> (as in the next step) to this service to be relayed to the  
453 identity provider, or it may rely on the intermediary service to issue an <AuthnRequest> message on its  
454 behalf.

### 455 4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider

456 Once an identity provider is selected, the location of its single sign-on service is determined, based on the  
457 SAML binding chosen by the service provider for sending the <AuthnRequest>. Metadata (as in  
458 [SAMLMeta]) MAY be used for this purpose. In response to an HTTP request by the user agent, an HTTP  
459 response is returned containing an <AuthnRequest> message or an artifact, depending on the SAML  
460 binding used, to be delivered to the identity provider's single sign-on service.

461 The exact format of this HTTP response and the subsequent HTTP request to the single sign-on service  
462 is defined by the SAML binding used. Profile-specific rules for the contents of the `<AuthnRequest>`  
463 message are included in Section 4.1.4.1. If the HTTP Redirect or POST binding is used, the  
464 `<AuthnRequest>` message is delivered directly to the identity provider in this step. If the HTTP Artifact  
465 binding is used, the Artifact Resolution profile defined in Section 5 is used by the identity provider, which  
466 makes a callback to the service provider to retrieve the `<AuthnRequest>` message, using, for example,  
467 the SOAP binding.

468 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or TLS  
469 1.0 [RFC2246] to maintain confidentiality and message integrity. The `<AuthnRequest>` message MAY  
470 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also  
471 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

472 The identity provider MUST process the `<AuthnRequest>` message as described in [SAMLCore]. This  
473 may constrain the subsequent interactions with the user agent, for example if the `IsPassive` attribute is  
474 included.

#### 475 **4.1.3.4 Identity Provider Identifies Principal**

476 At any time during the previous step or subsequent to it, the identity provider MUST establish the identity  
477 of the principal (unless it returns an error to the service provider). The `ForceAuthn` `<AuthnRequest>`  
478 attribute, if present with a value of `true`, obligates the identity provider to freshly establish this identity,  
479 rather than relying on an existing session it may have with the principal. Otherwise, and in all other  
480 respects, the identity provider may use any means to authenticate the user agent, subject to any  
481 requirements included in the `<AuthnRequest>` in the form of the `<RequestedAuthnContext>`  
482 element.

#### 483 **4.1.3.5 Identity Provider Issues `<Response>` to Service Provider**

484 Regardless of the success or failure of the `<AuthnRequest>`, the identity provider SHOULD produce an  
485 HTTP response to the user agent containing a `<Response>` message or an artifact, depending on the  
486 SAML binding used, to be delivered to the service provider's assertion consumer service.

487 The exact format of this HTTP response and the subsequent HTTP request to the assertion consumer  
488 service is defined by the SAML binding used. Profile-specific rules on the contents of the `<Response>`  
489 are included in Section 4.1.4.2. If the HTTP POST binding is used, the `<Response>` message is delivered  
490 directly to the service provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution  
491 profile defined in Section 5 is used by the service provider, which makes a callback to the identity provider  
492 to retrieve the `<Response>` message, using for example the SOAP binding.

493 The location of the assertion consumer service MAY be determined using metadata (as in [SAMLMeta]).  
494 The identity provider MUST have some means to establish that this location is in fact controlled by the  
495 service provider. A service provider MAY indicate the SAML binding and the specific assertion consumer  
496 service to use in its `<AuthnRequest>` and the identity provider MUST honor them if it can.

497 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 [SSL3] or TLS 1.0  
498 [RFC2246] to maintain confidentiality and message integrity. The `<Assertion>` element(s) in the  
499 `<Response>` MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-  
500 Artifact binding is used.

501 The service provider MUST process the `<Response>` message and any enclosed `<Assertion>`  
502 elements as described in [SAMLCore].

#### 503 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

504 To complete the profile, the service provider processes the `<Response>` and `<Assertion>`(s) and  
505 grants or denies access to the resource. The service provider MAY establish a security context with the

506 user agent using any session mechanism it chooses. Any subsequent use of the <Assertion>(s)  
507 provided are at the discretion of the service provider and other relying parties, subject to any restrictions  
508 on use contained within them.

#### 509 **4.1.4 Use of Authentication Request Protocol**

510 This profile is based on the Authentication Request protocol defined in [SAMLCore]. In the nomenclature  
511 of actors enumerated in Section 3.4 of that document, the service provider is the request issuer and the  
512 relying party, and the principal is the presenter, requested subject, and confirming entity. There may be  
513 additional relying parties or confirming entities at the discretion of the identity provider (see below).

##### 514 **4.1.4.1 <AuthnRequest> Usage**

515 A service provider MAY include any message content described in [SAMLCore], Section 3.4.1. All  
516 processing rules are as defined in [SAMLCore]. The <Issuer> element MUST be present and MUST  
517 contain the unique identifier of the requesting service provider; the Format attribute MUST be omitted or  
518 have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

519 If the identity provider cannot or will not satisfy the request, it MUST respond with a <Response>  
520 message containing an appropriate error status code or codes.

521 If the service provider wishes to permit the identity provider to establish a new identifier for the principal if  
522 none exists, it MUST include a <NameIDPolicy> element with the AllowCreate attribute set to "true".  
523 Otherwise, only a principal for whom the identity provider has previously established an identifier usable by  
524 the service provider can be authenticated successfully.

525 Note that the service provider MAY include a <Subject> element in the request that names the actual  
526 identity about which it wishes to receive an assertion. This element MUST NOT contain any  
527 <SubjectConfirmation> elements. If the identity provider does not recognize the principal as that  
528 identity, then it MUST respond with a <Response> message containing an error status and no assertions.

529 The <AuthnRequest> message MAY be signed (as directed by the SAML binding used). If the HTTP  
530 Artifact binding is used, authentication of the parties is OPTIONAL and any mechanism permitted by the  
531 binding MAY be used.

532 Note that if the <AuthnRequest> is not authenticated and/or integrity protected, the information in it  
533 MUST NOT be trusted except as advisory. Whether the request is signed or not, the identity provider  
534 MUST ensure that any <AssertionConsumerServiceURL> or  
535 <AssertionConsumerServiceIndex> elements in the request are verified as belonging to the service  
536 provider to whom the response will be sent. Failure to do so can result in a man-in-the-middle attack.

##### 537 **4.1.4.2 <Response> Usage**

538 If the identity provider wishes to return an error, it MUST NOT include any assertions in the <Response>  
539 message. Otherwise, if the request is successful (or if the response is not associated with a request), the  
540 <Response> element MUST conform to the following:

- 541 • The <Issuer> element MAY be omitted, but if present it MUST contain the unique identifier of the  
542 issuing identity provider; the Format attribute MUST be omitted or have a value of  
543 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 544 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the  
545 unique identifier of the issuing identity provider; the Format attribute MUST be omitted or have a value  
546 of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 547 • The set of one or more assertions MUST contain at least one <AuthnStatement> that reflects the  
548 authentication of the principal to the identity provider.

- 549 • At least one assertion containing an `<AuthnStatement>` MUST contain a `<Subject>` element with  
550 at least one `<SubjectConfirmation>` element containing a `Method` of  
551 `urn:oasis:names:tc:SAML:2.0:cm:bearer`. If the identity provider supports the Single Logout  
552 profile, defined in Section 4.4, any such authentication statements MUST include a `SessionIndex`  
553 attribute to enable per-session logout requests by the service provider.
- 554 • The bearer `<SubjectConfirmation>` element described above MUST contain a  
555 `<SubjectConfirmationData>` element that contains a `Recipient` attribute containing the service  
556 provider's assertion consumer service URL and a `NotOnOrAfter` attribute that limits the window  
557 during which the assertion can be delivered. It MAY contain an `Address` attribute limiting the client  
558 address from which the assertion can be delivered. It MUST NOT contain a `NotBefore` attribute. If  
559 the containing message is in response to an `<AuthnRequest>`, then the `InResponseTo` attribute  
560 MUST match the request's ID.
- 561 • Other statements and confirmation methods MAY be included in the assertion(s) at the discretion of  
562 the identity provider. In particular, `<AttributeStatement>` elements MAY be included. The  
563 `<AuthnRequest>` MAY contain an `AttributeConsumingServiceIndex` XML attribute  
564 referencing information about desired or required attributes in [SAMLMeta]. The identity provider MAY  
565 ignore this, or send other attributes at its discretion.
- 566 • The assertion(s) containing a bearer subject confirmation MUST contain an  
567 `<AudienceRestriction>` including the service provider's unique identifier as an `<Audience>`.
- 568 • Other conditions (and other `<Audience>` elements) MAY be included as requested by the service  
569 provider or at the discretion of the identity provider. (Of course, all such conditions MUST be  
570 understood by and accepted by the service provider in order for the assertion to be considered valid.)  
571 The identity provider is NOT obligated to honor the requested set of `<Conditions>` in the  
572 `<AuthnRequest>`, if any.

#### 573 4.1.4.3 `<Response>` Message Processing Rules

574 Regardless of the SAML binding used, the service provider MUST do the following:

- 575 • Verify any signatures present on the assertion(s) or the response
- 576 • Verify that the `Recipient` attribute in any bearer `<SubjectConfirmationData>` matches the  
577 assertion consumer service URL to which the `<Response>` or artifact was delivered
- 578 • Verify that the `NotOnOrAfter` attribute in any bearer `<SubjectConfirmationData>` has not  
579 passed, subject to allowable clock skew between the providers
- 580 • Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the ID  
581 of its original `<AuthnRequest>` message, unless the response is unsolicited (see Section 4.1.5 ), in  
582 which case the attribute MUST NOT be present
- 583 • Verify that any assertions relied upon are valid in other respects
- 584 • If any bearer `<SubjectConfirmationData>` includes an `Address` attribute, the service provider  
585 MAY check the user agent's client address against it.
- 586 • Any assertion which is not valid, or whose subject confirmation requirements cannot be met SHOULD  
587 be discarded and SHOULD NOT be used to establish a security context for the principal.
- 588 • If an `<AuthnStatement>` used to establish a security context for the principal contains a  
589 `SessionNotOnOrAfter` attribute, the security context SHOULD be discarded once this time is  
590 reached, unless the service provider reestablishes the principal's identity by repeating the use of this  
591 profile.

#### 592 **4.1.4.4 Artifact-Specific <Response> Message Processing Rules**

593 If the HTTP Artifact binding is used to deliver the <Response>, the dereferencing of the artifact using the  
594 Artifact Resolution profile MUST be mutually authenticated, integrity protected, and confidential.

595 The identity provider MUST ensure that only the service provider to whom the <Response> message has  
596 been issued is given the message as the result of an <ArtifactResolve> request.

597 Either the SAML binding used to dereference the artifact or message signatures can be used to  
598 authenticate the parties and protect the messages.

#### 599 **4.1.4.5 POST-Specific Processing Rules**

600 If the HTTP POST binding is used to deliver the <Response>, the enclosed assertion(s) MUST be  
601 signed.

602 The service provider MUST ensure that bearer assertions are not replayed, by maintaining the set of used  
603 ID values for the length of time for which the assertion would be considered valid based on the  
604 NotOnOrAfter attribute in the <SubjectConfirmationData>.

#### 605 **4.1.5 Unsolicited Responses**

606 An identity provider MAY initiate this profile by delivering an unsolicited <Response> message to a  
607 service provider.

608 An unsolicited <Response> MUST NOT contain an InResponseTo attribute, nor should any bearer  
609 <SubjectConfirmationData> elements contain one. If metadata as specified in [SAMLMeta] is used,  
610 the <Response> or artifact SHOULD be delivered to the <md:AssertionConsumerService> endpoint  
611 of the service provider designated as the default.

612 Of special mention is that the identity provider MAY include a binding-specific "RelayState" parameter that  
613 indicates, based on mutual agreement with the service provider, how to handle subsequent interactions  
614 with the user agent. This MAY be the URL of a resource at the service provider. The service provider  
615 SHOULD be prepared to handle unsolicited responses by designating a default location to send the user  
616 agent subsequent to processing a response successfully.

#### 617 **4.1.6 Use of Metadata**

618 [SAMLMeta] defines an endpoint element, <md:SingleSignOnService>, to describe supported  
619 bindings and location(s) to which a service provider may send requests to an identity provider using this  
620 profile.

621 The <md:IDPSSODescriptor> element's WantAuthnRequestsSigned attribute MAY be used by an  
622 identity provider to document a requirement that requests be signed. The <md:SPSSODescriptor>  
623 element's AuthnRequestsSigned attribute MAY be used by a service provider to document the  
624 intention to sign all of its requests.

625 The providers MAY document the key(s) used to sign requests, responses, and assertions with  
626 <md:KeyDescriptor> elements with a use attribute of sign. When encrypting SAML elements,  
627 <md:KeyDescriptor> elements with a use attribute of encrypt MAY be used to document supported  
628 encryption algorithms and settings, and public keys used to receive bulk encryption keys.

629 The indexed endpoint element <md:AssertionConsumerService> is used to describe supported  
630 bindings and location(s) to which an identity provider may send responses to a service provider using this  
631 profile. The index attribute is used to distinguish the possible endpoints that may be specified by  
632 reference in the <AuthnRequest> message. The isDefault attribute is used to specify the endpoint to  
633 use if not specified in a request.

634 The `<md:SPSSODescriptor>` element's `WantAssertionsSigned` attribute MAY be used by a service  
635 provider to document a requirement that assertions delivered with this profile be signed. This is in addition  
636 to any requirements for signing imposed by the use of a particular binding. Note that the identity provider  
637 is not obligated by this, but is being made aware of the likelihood that an unsigned assertion will be  
638 insufficient.

639 If the request or response message is delivered using the HTTP Artifact binding, the artifact issuer MUST  
640 provide at least one `<md:ArtifactResolutionService>` endpoint element in its metadata.

641 The `<md:IDPSSODescriptor>` MAY contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and  
642 `<saml:Attribute>` elements to indicate the general ability to support particular name identifier formats,  
643 attribute profiles, or specific attributes and values. The ability to support any such features during a given  
644 authentication exchange is dependent on policy and the discretion of the identity provider.

645 The `<md:SPSSODescriptor>` element MAY also be used to document the service provider's need or  
646 desire for SAML attributes to be delivered along with authentication information. The actual inclusion of  
647 attributes is always at the discretion of the identity provider. One or more  
648 `<md:AttributeConsumingService>` elements MAY be included in its metadata, each with an `index`  
649 attribute to distinguish different services that MAY be specified by reference in the `<AuthnRequest>`  
650 message. The `isDefault` attribute is used to specify a default set of attribute requirements.

## 651 4.2 Enhanced Client or Proxy (ECP) Profile

652 An *enhanced client or proxy* (ECP) is a system entity that knows how to contact an appropriate identity  
653 provider, possibly in a context-dependent fashion, and also supports the Reverse SOAP (PAOS) binding  
654 [SAMLBind].

655 An example scenario enabled by this profile is as follows: A principal, wielding an ECP, uses it to either  
656 access a resource at a service provider, or access an identity provider such that the service provider and  
657 desired resource are understood or implicit. The principal authenticates (or has already authenticated)  
658 with the identity provider, which then produces an authentication assertion (possibly with input from the  
659 service provider). The service provider then consumes the assertion and subsequently establishes a  
660 security context for the principal. During this process, a name identifier might also be established between  
661 the providers for the principal, subject to the parameters of the interaction and the consent of the principal.

662 This profile is based on the SAML Authentication Request protocol [SAMLCore] in conjunction with the  
663 PAOS binding.

664 **Note:** The means by which a principal authenticates with an identity provider is outside of the  
665 scope of SAML.

### 666 4.2.1 Required Information

667 **Identification:** `urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp` (this is also the target namespace  
668 assigned in the corresponding ECP profile schema document [SAMLECP-xsd])

669 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

670 **SAML Confirmation Method Identifiers:** The SAML V2.0 "bearer" confirmation method identifier,  
671 `urn:oasis:names:tc:SAML:2.0:cm:bearer`, is used by this profile.

672 **Description:** Given below.

673 **Updates:** None.

### 674 4.2.2 Profile Overview

675 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and

676 service providers and identity providers. It is a specific application of the SSO profile described in Section  
677 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the  
678 rules specified in Section 4.1 MUST be observed.

679 An ECP is a client or proxy that satisfies the following two conditions:

- 680 • It has, or knows how to obtain, information about the identity provider that the principal associated with  
681 the ECP wishes to use, in the context of an interaction with a service provider.

682 This allows a service provider to make an authentication request to the ECP without the need to know  
683 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section  
684 4.1).

- 685 • It is able to use a reverse SOAP (PAOS) binding as profiled here for an authentication request and  
686 response.

687 This enables a service provider to obtain an authentication assertion via an ECP that is not otherwise  
688 (i.e. outside of the context of the immediate interaction) necessarily directly addressable nor  
689 continuously available. It also leverages the benefits of SOAP while using a well-defined exchange  
690 pattern and profile to enable interoperability. The ECP may be viewed as a SOAP intermediary  
691 between the service provider and the identity provider.

692 An *enhanced client* may be a browser or some other user agent that supports the functionality described  
693 in this profile. An *enhanced proxy* is an HTTP proxy (for example a WAP gateway) that emulates an  
694 enhanced client. Unless stated otherwise, all statements referring to enhanced clients are to be  
695 understood as statements about both enhanced clients as well as enhanced client proxies.

696 Since the enhanced client sends and receives messages in the body of HTTP requests and responses, it  
697 has no arbitrary restrictions on the size of the protocol messages.

698 This profile leverages the Reverse SOAP (PAOS) binding [SAMLBind]. Implementers of this profile MUST  
699 follow the rules for HTTP indications of PAOS support specified in that binding, in addition to those  
700 specified in this profile. This profile utilizes a PAOS SOAP header block conveyed between the HTTP  
701 responder and the ECP but does not define PAOS itself. The SAML PAOS binding specification  
702 [SAMLBind] is normative in the event of questions regarding PAOS.

703 This profile defines SOAP header blocks that accompany the SAML requests and responses. These  
704 header blocks may be composed with other SOAP header blocks as necessary, for example with the  
705 SOAP Message Security header block to add security features if needed, for example a digital signature  
706 applied to the authentication request.

707 Two sets of request/response SOAP header blocks are used: PAOS header blocks for generic PAOS  
708 information and ECP profile-specific header blocks to convey information specific to ECP profile  
709 functionality.

710 Figure 2 shows the processing flow in the ECP profile.

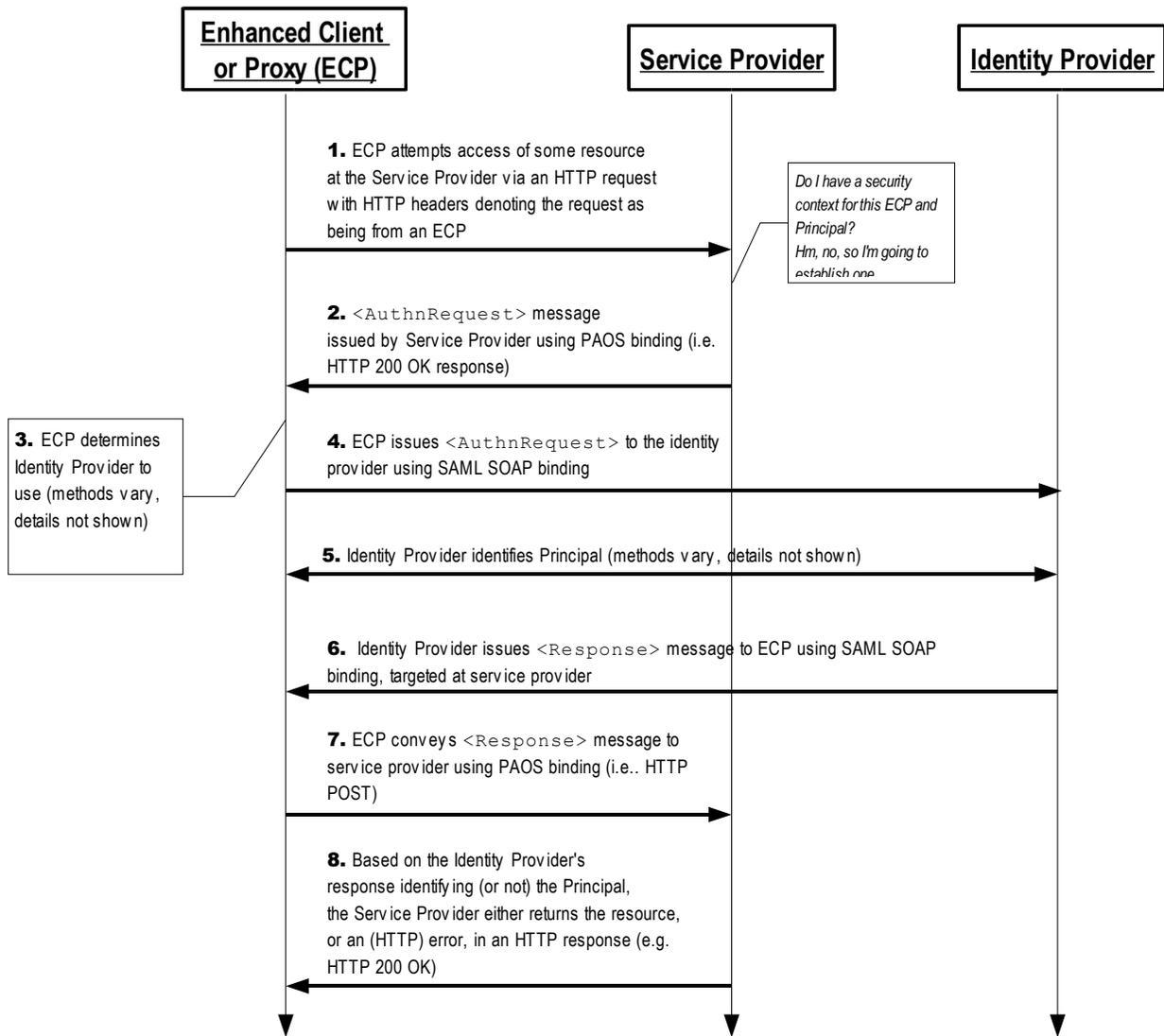


Figure 2

711 Figure 2 illustrates the basic template for SSO using an ECP. The following steps are described by the  
 712 profile. Within an individual step, there may be one or more actual message exchanges depending on the  
 713 binding used for that step and other implementation-dependent behavior.

714 **1. ECP issues HTTP Request to Service Provider**

715 In step 1, the Principal, via an ECP, makes an HTTP request for a secured resource at a service  
 716 provider, where the service provider does not have an established security context for the ECP  
 717 and Principal.

718 **2. Service Provider issues <AuthnRequest> to ECP**

719 In step 2, the service provider issues an <AuthnRequest> message to the ECP, which is to be  
 720 delivered by the ECP to the appropriate identity provider. The Reverse SOAP (PAOS) binding  
 721 [SAMLBind] is used here.

722 **3. ECP Determines Identity Provider**

723 In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication  
724 request protocol that supports its preferred binding. The means by which this is accomplished is  
725 implementation-dependent. The ECP MAY use the SAML identity provider discovery profile  
726 described in Section 4.3.

#### 727 **4. ECP conveys <AuthnRequest> to Identity Provider**

728 In step 4, the ECP conveys the <AuthnRequest> to the identity provider identified in step 3  
729 using a modified form of the SAML SOAP binding [SAMLBind] with the additional allowance that  
730 the identity provider may exchange arbitrary HTTP messages with the ECP before responding to  
731 the SAML request.

#### 732 **5. Identity Provider identifies Principal**

733 In step 5, the Principal is identified by the identity provider by some means outside the scope of  
734 this profile. This may require a new act of authentication, or it may reuse an existing authenticated  
735 session.

#### 736 **6. Identity Provider issues <Response> to ECP, targeted at Service Provider**

737 In step 6, the identity provider issues a <Response> message, using the SAML SOAP binding, to  
738 be delivered by the ECP to the service provider. The message may indicate an error, or will  
739 include (at least) an authentication assertion.

#### 740 **7. ECP conveys <Response> message to Service Provider**

741 In step 7, the ECP conveys the <Response> message to the service provider using the PAOS  
742 binding.

#### 743 **8. Service Provider grants or denies access to Principal**

744 In step 8, having received the <Response> message from the identity provider, the service  
745 provider either establishes its own security context for the principal and return the requested  
746 resource, or responds to the principal's ECP with an error.

### 747 **4.2.3 Profile Description**

748 The following sections provide detailed definitions of the individual steps.

#### 749 **4.2.3.1 ECP issues HTTP Request to Service Provider**

750 The ECP sends an HTTP request to a service provider, specifying some resource. This HTTP request  
751 MUST conform to the PAOS binding, which means it must include the following HTTP header fields:

- 752 1. The HTTP Accept Header field indicating the ability to accept the MIME type  
753 "application/vnd.paos+xml"
- 754 2. The HTTP PAOS Header field specifying the PAOS version with urn:liberty:paos:2003-08 at  
755 minimum.
- 756 3. Furthermore, support for this profile MUST be specified in the HTTP PAOS Header field as a service  
757 value, with the value urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp. This value should  
758 correspond to the service attribute in the PAOS Request SOAP header block

759 For example, a user agent may request a page from a service provider as follows:

```
760 GET /index HTTP/1.1  
761 Host: identity-service.example.com  
762 Accept: text/html; application/vnd.paos+xml  
763 PAOS: ver='urn:liberty:paos:2003-08' ;  
764 'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
```

#### 765 **4.2.3.2 Service Provider Issues <AuthnRequest> to ECP**

766 When the service provider requires a security context for the principal before allowing access to the  
767 specified resource, that is, before providing a service or data, it can respond to the HTTP request using  
768 the PAOS binding with an <AuthnRequest> message in the HTTP response. The service provider will  
769 issue an HTTP 200 OK response to the ECP containing a single SOAP envelope.

770 The SOAP envelope MUST contain:

- 771 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the  
772 identity provider.
- 773 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of  
774 `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control  
775 information such as the URL to which to send the response in this solicit-response message  
776 exchange pattern.
- 777 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor  
778 `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines  
779 information related to the authentication request that the ECP may need to process it, such as a list  
780 of identity providers acceptable to the service provider, whether the ECP may interact with the  
781 principal through the client, and the service provider's human-readable name that may be displayed  
782 to the principal.

783 The SOAP envelope MAY contain an ECP RelayState SOAP header block targeted at the ECP using the  
784 SOAP actor value of `http://schemas.xmlsoap.org/soap/actor/next`. The header contains state information  
785 to be returned by the ECP along with the SAML response.

#### 786 **4.2.3.3 ECP Determines Identity Provider**

787 The ECP will determine which identity provider is appropriate and route the SOAP message appropriately.

#### 788 **4.2.3.4 ECP issues <AuthnRequest> to Identity Provider**

789 The ECP MUST remove the PAOS, ECP RelayState, and ECP Request header blocks before passing the  
790 <AuthnRequest> message on to the identity provider, using a modified form of the SAML SOAP binding.  
791 The SAML request is submitted via SOAP in the usual fashion, but the identity provider MAY respond to  
792 the ECP's HTTP request with an HTTP response containing, for example, an HTML login form or some  
793 other presentation-oriented response. A sequence of HTTP exchanges MAY take place, but ultimately the  
794 identity provider MUST complete the SAML SOAP exchange and return a SAML response via the SOAP  
795 binding.

796 Note that the <AuthnRequest> element may itself be signed by the service provider. In this and other  
797 respects, the message rules specified in the browser SSO profile in Section 4.1.4.1 MUST be followed.

798 Prior to or subsequent to this step, the identity provider MUST establish the identity of the principal by  
799 some means, or it MUST return an error <Response>, as described in Section 4.2.3.6 below.

#### 800 **4.2.3.5 Identity Provider Identifies Principal**

801 At any time during the previous step or subsequent to it, the identity provider MUST establish the identity  
802 of the principal (unless it returns an error to the service provider). The `ForceAuthn` <AuthnRequest>  
803 attribute, if present with a value of `true`, obligates the identity provider to freshly establish this identity,  
804 rather than relying on an existing session it may have with the principal. Otherwise, and in all other  
805 respects, the identity provider may use any means to authenticate the user agent, subject to any  
806 requirements included in the <AuthnRequest> in the form of the <RequestedAuthnContext>  
807 element.

#### 808 4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider

809 The identity provider returns a SAML <Response> message (or SOAP fault) when presented with an  
810 authentication request, after having established the identity of the principal. The SAML response is  
811 conveyed using the SAML SOAP binding in a SOAP message with a <Response> element in the SOAP  
812 body, intended for the service provider as the ultimate SOAP receiver. The rules for the response  
813 specified in the browser SSO profile in Section 4.1.4.2 MUST be followed.

814 The identity provider's response message MUST contain a profile-specific ECP Response SOAP header  
815 block, and MAY contain an ECP RelayState header block, both targeted at the ECP.

#### 816 4.2.3.7 ECP Conveys <Response> Message to Service Provider

817 The ECP removes the header block(s), and MAY add a PAOS Response SOAP header block and an  
818 ECP RelayState header block before forwarding the SOAP response to the service provider using the  
819 PAOS binding.

820 The <paos:Response> SOAP header block in the response to the service provider is generally used to  
821 correlate this response to an earlier request from the service provider. In this profile, the correlation  
822 refToMessageID attribute is not required since the SAML <Response> element's InResponseTo  
823 attribute may be used for this purpose, but if the <paos:Request> SOAP Header block had a  
824 messageID then the <paos:Response> SOAP header block MUST be used.

825 The <ecp:RelayState> header block value is typically provided by the service provider to the ECP with  
826 its request, but if the identity provider is producing an unsolicited response (without having received a  
827 corresponding SAML request), then it MAY include a RelayState header block that indicates, based on  
828 mutual agreement with the service provider, how to handle subsequent interactions with the ECP. This  
829 MAY be the URL of a resource at the service provider.

830 If the service provider included an <ecp:RelayState> SOAP header block in its request to the ECP, or  
831 if the identity provider included an <ecp:RelayState> SOAP header block with its response, then the  
832 ECP MUST include an identical header block with the SAML response sent to the service provider. The  
833 service provider's value for this header block (if any) MUST take precedence.

#### 834 4.2.3.8 Service Provider Grants or Denies Access to Principal

835 Once the service provider has received the SAML response in an HTTP request (in a SOAP envelope  
836 using PAOS), it may respond with the service data in the HTTP response. In consuming the response, the  
837 rules specified in the browser SSO profile in Section 4.1.4.3 and 4.1.4.5 MUST be followed. That is, the  
838 same processing rules used when receiving the <Response> with the HTTP POST binding apply to the  
839 use of PAOS.

#### 840 4.2.4 ECP Profile Schema Usage

841 The ECP Profile XML schema [SAMLECP-xsd] defines the SOAP Request/Response header blocks used  
842 by this profile. Following is a complete listing of this schema document.

```
843 <schema  
844   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
845   xmlns="http://www.w3.org/2001/XMLSchema"  
846   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
847   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
848   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
849   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"  
850   elementFormDefault="unqualified"  
851   attributeFormDefault="unqualified"  
852   blockDefault="substitution"  
853   version="2.0">
```

```

854 <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
855       schemaLocation="saml-schema-protocol-2.0.xsd"/>
856 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
857       schemaLocation="saml-schema-assertion-2.0.xsd"/>
858 <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
859       schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
860 <annotation>
861   <documentation>
862     Document identifier: saml-schema-ecp-2.0
863     Location: http://docs.oasis-open.org/security/saml/v2.0/
864     Revision history:
865       V2.0 (March, 2005):
866         Custom schema for ECP profile, first published in SAML 2.0.
867   </documentation>
868 </annotation>

869 <element name="Request" type="ecp:RequestType"/>
870 <complexType name="RequestType">
871   <sequence>
872     <element ref="saml:Issuer"/>
873     <element ref="samlp:IDPList" minOccurs="0"/>
874   </sequence>
875   <attribute ref="S:mustUnderstand" use="required"/>
876   <attribute ref="S:actor" use="required"/>
877   <attribute name="ProviderName" type="string" use="optional"/>
878   <attribute name="IsPassive" type="boolean" use="optional"/>
879 </complexType>

880 <element name="Response" type="ecp:ResponseType"/>
881 <complexType name="ResponseType">
882   <attribute ref="S:mustUnderstand" use="required"/>
883   <attribute ref="S:actor" use="required"/>
884   <attribute name="AssertionConsumerServiceURL" type="anyURI"
885 use="required"/>
886 </complexType>

887 <element name="RelayState" type="ecp:RelayStateType"/>
888 <complexType name="RelayStateType">
889   <simpleContent>
890     <extension base="string">
891       <attribute ref="S:mustUnderstand" use="required"/>
892       <attribute ref="S:actor" use="required"/>
893     </extension>
894   </simpleContent>
895 </complexType>
896 </schema>

```

899 The following sections describe how these XML constructs are to be used.

#### 900 **4.2.4.1 PAOS Request Header Block: SP to ECP**

901 The PAOS Request header block signals the use of PAOS processing and includes the following  
902 attributes:

903 responseConsumerURL [Required]

904 Specifies where the ECP is to send an error response. Also used to verify the correctness of the  
905 identity provider's response, by cross checking this location against the  
906 AssertionServiceConsumerURL in the ECP response header block. This value MUST be the  
907 same as the AssertionServiceConsumerURL (or the URL referenced in metadata) conveyed in  
908 the <AuthnRequest>.

909 service [Required]

910 Indicates that the PAOS service being used is this SAML authentication profile. The value MUST be  
911 urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp.

912 SOAP-ENV:mustUnderstand [Required]  
913 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not  
914 understood.

915 SOAP-ENV:actor [Required]  
916 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

917 messageID [Optional]  
918 Allows optional response correlation. It MAY be used in this profile, but is NOT required, since this  
919 functionality is provided by the SAML protocol layer, via the ID attribute in the <AuthnRequest> and  
920 the InResponseTo attribute in the <Response>.

921 The PAOS Request SOAP header block has no element content.

#### 922 4.2.4.2 ECP Request Header Block: SP to ECP

923 The ECP Request SOAP header block is used to convey information needed by the ECP to process the  
924 authentication request. It is mandatory and its presence signals the use of this profile. It contains the  
925 following elements and attributes:

926 SOAP-ENV:mustUnderstand [Required]  
927 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not  
928 understood.

929 SOAP-ENV:actor [Required]  
930 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

931 ProviderName [Optional]  
932 A human-readable name for the requesting service provider.

933 IsPassive [Optional]  
934 A boolean value. If true, the identity provider and the client itself MUST NOT take control of the user  
935 interface from the request issuer and interact with the principal in a noticeable fashion. If a value is not  
936 provided, the default is true.

937 <saml:Issuer> [Required]  
938 This element MUST contain the unique identifier of the requesting service provider; the Format  
939 attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
940 `format:entity`.

941 <samlp:IDPList> [Optional]  
942 Optional list of identity providers that the service provider recognizes and from which the ECP may  
943 choose to service the request. See [SAMLCore] for details on the content of this element.

#### 944 4.2.4.3 ECP RelayState Header Block: SP to ECP

945 The ECP RelayState SOAP header block is used to convey state information from the service provider  
946 that it will need later when processing the response from the ECP. It is optional, but if used, the ECP  
947 MUST include an identical header block in the response in step 5. It contains the following attributes:

948 SOAP-ENV:mustUnderstand [Required]  
949 The value MUST be 1 (true). A SOAP fault MUST be generated if the header block is not understood.

950 SOAP-ENV:actor [Required]

951 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

952 The content of the header block element is a string containing state information created by the requester.  
953 If provided, the ECP MUST include the same value in a RelayState header block when responding to the  
954 service provider in step 5. The string value MUST NOT exceed 80 bytes in length and SHOULD be  
955 integrity protected by the requester independent of any other protections that may or may not exist during  
956 message transmission.

957 The following is an example of the SOAP authentication request from the service provider to the ECP:

```
958 <SOAP-ENV:Envelope
959     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
960     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
961     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
962   <SOAP-ENV:Header>
963     <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
964         responseConsumerURL="http://identity-service.example.com/abc"
965         messageID="6c3a4f8b9c2d" SOAP-
966     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
967     ENV:mustUnderstand="1"
968         service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
969       </paos:Request>
970       <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
971         SOAP-ENV:mustUnderstand="1" SOAP-
972     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
973         ProviderName="Service Provider X" IsPassive="0">
974       <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
975       <samlp:IDPList>
976         <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
977             Name="Identity Provider X"
978             Loc="https://IdentityProvider.example.com/saml2/sso"
979         </samlp:IDPEntry>
980         <samlp:GetComplete>
981           https://ServiceProvider.example.com/idplist?id=604bel36-fe91-441e-afb8
982         </samlp:GetComplete>
983       </samlp:IDPList>
984     </ecp:Request>
985     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
986         SOAP-ENV:mustUnderstand="1" SOAP-
987     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
988       ...
989     </ecp:RelayState>
990   </SOAP-ENV:Header>
991   <SOAP-ENV:Body>
992     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
993   </SOAP-ENV:Body>
994 </SOAP-ENV:Envelope>
```

995 As noted above, the PAOS and ECP header blocks are removed from the SOAP message by the ECP  
996 before the authentication request is forwarded to the identity provider. An example authentication request  
997 from the ECP to the identity provider is as follows:

```
998 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
999     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
1000   <SOAP-ENV:Body>
1001     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1002   </SOAP-ENV:Body>
1003 </SOAP-ENV:Envelope>
```

#### 1004 4.2.4.4 ECP Response Header Block: IdP to ECP

1005 The ECP response SOAP header block MUST be used on the response from the identity provider to the  
1006 ECP. It contains the following attributes:

1007 SOAP-ENV:mustUnderstand [Required]

1008 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not  
1009 understood.

1010 SOAP-ENV:actor [Required]

1011 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1012 AssertionConsumerServiceURL [Required]

1013 Set by the identity provider based on the <AuthnRequest> message or the service provider's  
1014 metadata obtained by the identity provider.

1015 The ECP MUST confirm that this value corresponds to the value the ECP obtained in the  
1016 responseConsumerURL in the PAOS Request SOAP header block it received from the service  
1017 provider. Since the responseConsumerURL MAY be relative and the  
1018 AssertionConsumerServiceURL is absolute, some processing/normalization may be required.

1019 This mechanism is used for security purposes to confirm the correct response destination. If the  
1020 values do not match, then the ECP MUST generate a SOAP fault response to the service provider  
1021 and MUST NOT return the SAML response.

1022 The ECP Response SOAP header has no element content.

1023 Following is an example of an IdP-to-ECP response.

```
1024 <SOAP-ENV:Envelope  
1025     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
1026     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1027     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  
1028   <SOAP-ENV:Header>  
1029     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-  
1030     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"  
1031     AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion  
1032     consumer"/>  
1033   </SOAP-ENV:Header>  
1034   <SOAP-ENV:Body>  
1035     <samlp:Response> ... </samlp:Response>  
1036   </SOAP-ENV:Body>  
1037 </SOAP-ENV:Envelope>
```

#### 1038 4.2.4.5 PAOS Response Header Block: ECP to SP

1039 The PAOS Response header block includes the following attributes:

1040 SOAP-ENV:mustUnderstand [Required]

1041 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not  
1042 understood.

1043 SOAP-ENV:actor [Required]

1044 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1045 refToMessageID [Optional]

1046 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)  
1047 MUST be added by the ECP if the corresponding PAOS request specified the messageID attribute.  
1048 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>  
1049 correlation.

1050 The PAOS Response SOAP header has no element content.

1051 Following is an example of an ECP-to-SP response.

```
1052 <SOAP-ENV:Envelope  
1053     xmlns:paos="urn:liberty:paos:2003-08"
```

```

1054     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1055     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
1056     <SOAP-ENV:Header>
1057         <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1058     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1059     ENV:mustUnderstand="1"/>
1060         <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1061     SOAP-ENV:mustUnderstand="1" SOAP-
1062     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1063             ...
1064         </ecp:RelayState>
1065     </SOAP-ENV:Header>
1066     <SOAP-ENV:Body>
1067         <samlp:Response> ... </samlp:Response>
1068     </SOAP-ENV:Body>
1069 </SOAP-ENV:Envelope>

```

## 1070 4.2.5 Security Considerations

1071 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,  
 1072 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP  
 1073 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security  
 1074 countermeasures appropriate to that binding are used.

1075 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the  
 1076 use of SSL/TLS over every HTTP exchange with the client.

1077 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS  
 1078 authentication.

1079 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated  
 1080 session. Any HTTP exchanges subsequent to the delivery of the <AuthnRequest> message and before  
 1081 the identity provider returns a <Response> MUST be securely associated with the original request.

## 1082 4.3 Identity Provider Discovery Profile

1083 This section defines a profile by which a service provider can discover which identity providers a principal  
 1084 is using with the Web Browser SSO profile. In deployments having more than one identity provider,  
 1085 service providers need a means to discover which identity provider(s) a principal uses. The discovery  
 1086 profile relies on a cookie that is written in a domain that is common between identity providers and service  
 1087 providers in a deployment. The domain that the deployment predetermines is known as the common  
 1088 domain in this profile, and the cookie containing the list of identity providers is known as the common  
 1089 domain cookie.

1090 Which entities host web servers in the common domain is a deployment issue and is outside the scope of  
 1091 this profile.

### 1092 4.3.1 Common Domain Cookie

1093 The name of the cookie MUST be "\_saml\_idp". The format of the cookie value MUST be a set of one or  
 1094 more base-64 encoded URI values separated by a single space character. Each URI is the unique  
 1095 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then  
 1096 URL encoded.

1097 The common domain cookie writing service (see below) SHOULD append the identity provider's unique  
 1098 identifier to the list. If the identifier is already present in the list, it MAY remove and append it. The intent is  
 1099 that the most recently established identity provider session is the last one in the list.

1100 The cookie MUST be set with a Path prefix of "/". The Domain MUST be set to "[common-domain]" where  
 1101 [common-domain] is the common domain established within the deployment for use with this profile.

1102 There MUST be a leading period. The cookie MUST be marked as secure.

1103 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY  
1104 be either session-only or persistent. This choice may be made within a deployment, but should apply  
1105 uniformly to all identity providers in the deployment.

### 1106 **4.3.2 Setting the Common Domain Cookie**

1107 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by  
1108 which the identity provider sets the cookie are implementation-specific so long as the cookie is  
1109 successfully set with the parameters given above. One possible implementation strategy follows and  
1110 should be considered non-normative. The identity provider may:

- 1111 • Have previously established a DNS and IP alias for itself in the common domain.
- 1112 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL  
1113 scheme. The structure of the URL is private to the implementation and may include session  
1114 information needed to identify the user agent.
- 1115 • Set the cookie on the redirected user agent using the parameters specified above.
- 1116 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

### 1117 **4.3.3 Obtaining the Common Domain Cookie**

1118 When a service provider needs to discover which identity providers a principal uses, it invokes an  
1119 exchange designed to present the common domain cookie to the service provider after it is read by an  
1120 HTTP server in the common domain.

1121 If the HTTP server in the common domain is operated by the service provider or if other arrangements are  
1122 in place, the service provider MAY utilize the HTTP server in the common domain to relay its  
1123 <AuthnRequest> to the identity provider for an optimized single sign-on process.

1124 The specific means by which the service provider reads the cookie are implementation-specific so long as  
1125 it is able to cause the user agent to present cookies that have been set with the parameters given in  
1126 Section 4.3.1. One possible implementation strategy is described as follows and should be considered  
1127 non-normative. Additionally, it may be sub-optimal for some applications.

- 1128 • Have previously established a DNS and IP alias for itself in the common domain.
- 1129 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL  
1130 scheme. The structure of the URL is private to the implementation and may include session  
1131 information needed to identify the user agent.
- 1132 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

## 1133 **4.4 Single Logout Profile**

1134 Once a principal has authenticated to an identity provider, the authenticating entity may establish a  
1135 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-  
1136 specific means). The identity provider may subsequently issue assertions to service providers or other  
1137 relying parties, based on this authentication event; a relying party may use this to establish *its own* session  
1138 with the principal.

1139 In such a situation, the identity provider can act as a session authority and the relying parties as session  
1140 participants. At some later time, the principal may wish to terminate his or her session either with an  
1141 individual session participant, or with all session participants in a given session managed by the session  
1142 authority. The former case is considered out of scope of this specification. The latter case, however, may  
1143 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1144 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this  
1145 "global" session either by contacting the session authority, or an individual session participant. Also note  
1146 that an identity provider acting as a session authority may *itself* act as a session participant in situations in  
1147 which it is the relying party for another identity provider's assertions regarding that principal.

1148 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or  
1149 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A  
1150 front-channel binding may be required, for example, in cases in which a principal's session state exists  
1151 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the  
1152 session participant or session authority is required. As will be discussed below, session participants  
1153 should if possible use a "front-channel" binding when initiating this profile to maximize the likelihood that  
1154 the session authority can propagate the logout successfully to all participants.

#### 1155 4.4.1 Required Information

1156 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1157 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1158 **Description:** Given below.

1159 **Updates:** None

#### 1160 4.4.2 Profile Overview

1161 Figure 3 illustrates the basic template for achieving single logout:

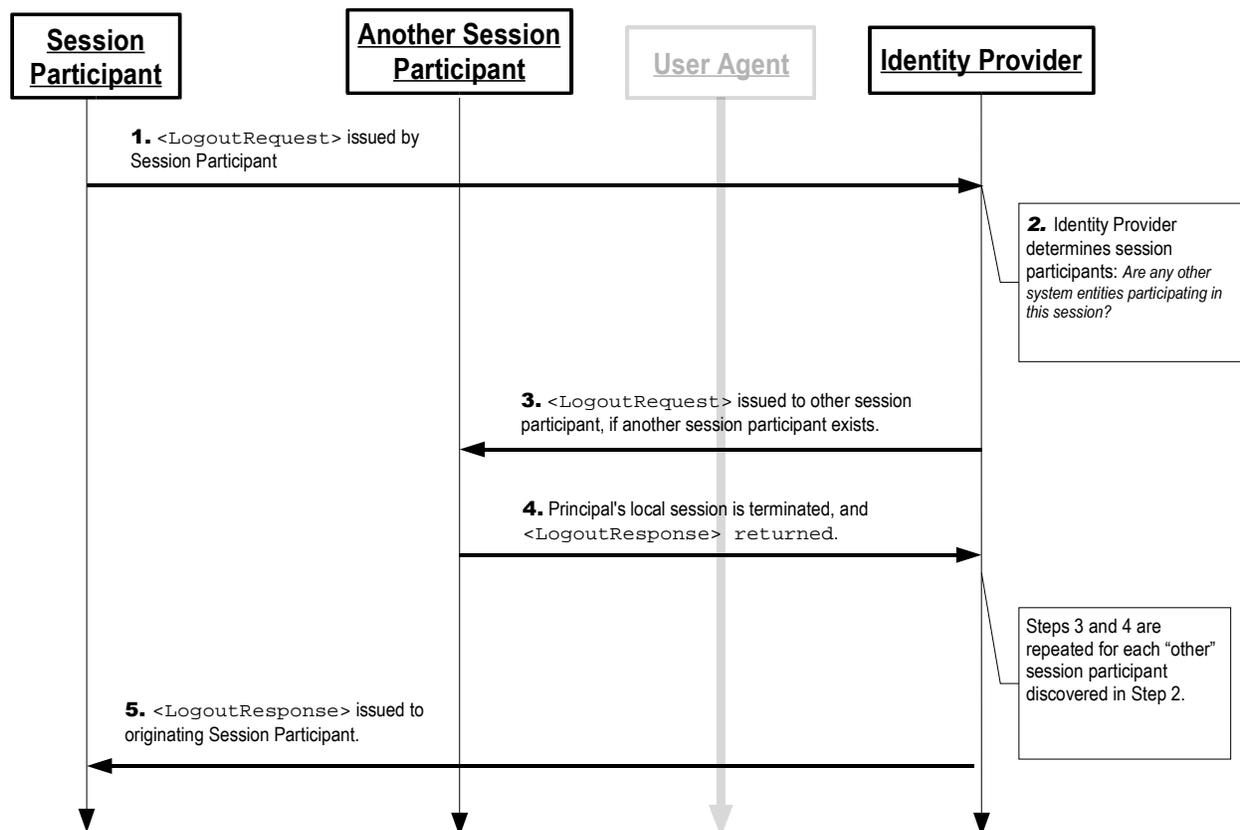


Figure 3

1162 The grayed-out user agent illustrates that the message exchange may pass through the user agent or  
1163 may be a direct exchange between system entities, depending on the SAML binding used to implement  
1164 the profile.

1165 The following steps are described by the profile. Within an individual step, there may be one or more  
1166 actual message exchanges depending on the binding used for that step and other implementation-  
1167 dependent behavior.

#### 1168 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1169 In step 1, the session participant initiates single logout and terminates a principal's session(s) by  
1170 sending a <LogoutRequest> message to the identity provider from whom it received the  
1171 corresponding authentication assertion. The request may be sent directly to the identity provider  
1172 or sent indirectly through the user agent.

#### 1173 **2. Identity Provider determines Session Participants**

1174 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if  
1175 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If  
1176 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4  
1177 are repeated for each session participant identified.

#### 1178 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1179 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or  
1180 session authority related to one or more of the session(s) being terminated. The request may be  
1181 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the  
1182 request in step 1).

#### 1183 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1184 In step 4, a session participant or session authority terminates the principal's session(s) as  
1185 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.  
1186 The response may be returned directly to the identity provider or indirectly through the user agent  
1187 (if consistent with the form of the request in step 3).

#### 1188 **5. Identity Provider issues <LogoutResponse> to Session Participant**

1189 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting  
1190 session participant. The response may be returned directly to the session participant or indirectly  
1191 through the user agent (if consistent with the form of the request in step 1).

1192 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a  
1193 <LogoutRequest> to all session participants, also skipping step 5.

### 1194 **4.4.3 Profile Description**

1195 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity  
1196 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

#### 1197 **Single Logout Service**

1198 This is the single logout protocol endpoint at an identity provider or session participant to which the  
1199 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are  
1200 delivered. The same or different endpoints MAY be used for requests and responses.

#### 1201 **4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider**

1202 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it  
1203 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it

1204 received from the identity provider. If multiple identity providers are involved, then the profile MUST be  
1205 repeated independently for each one.

1206 To initiate the profile, the session participant issues a <LogoutRequest> message to the identity  
1207 provider's single logout service request endpoint containing one or more applicable <SessionIndex>  
1208 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to  
1209 determine the location of this endpoint and the bindings supported by the identity provider.

#### 1210 **Asynchronous Bindings (Front-Channel)**

1211 The session participant SHOULD (if the principal's user agent is present) use an asynchronous  
1212 binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind], to send the request to  
1213 the identity provider through the user agent. The identity provider SHOULD then propagate any  
1214 required logout messages to additional session participants as required using either a synchronous or  
1215 asynchronous binding. The use of an asynchronous binding for the original request is preferred  
1216 because it gives the identity provider the best chance of successfully propagating the logout to the  
1217 other session participants during step 3.

1218 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is delivered to  
1219 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile  
1220 defined in Section 5 is used by the identity provider, which makes a callback to the session participant  
1221 to retrieve the <LogoutRequest> message, using for example the SOAP binding.

1222 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or  
1223 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The <LogoutRequest>  
1224 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,  
1225 if used, also provides for an alternate means of authenticating the request issuer when the artifact is  
1226 dereferenced.

1227 Each of these bindings provide a RelayState mechanism that the session participant MAY use to  
1228 associate the profile exchange with the original request. The session participant SHOULD reveal as  
1229 little information as possible in the RelayState value unless the use of the profile does not require such  
1230 privacy measures.

#### 1231 **Synchronous Bindings (Back-Channel)**

1232 Alternatively, the session participant MAY use a synchronous binding, such as the SOAP binding  
1233 [SAMLBind], to send the request directly to the identity provider. The identity provider SHOULD then  
1234 propagate any required logout messages to additional session participants as required using a  
1235 synchronous binding. The requester MUST authenticate itself to the identity provider, either by signing  
1236 the <LogoutRequest> or using any other binding-supported mechanism.

1237 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

### 1238 **4.4.3.2 Identity Provider Determines Session Participants**

1239 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>  
1240 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the  
1241 identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1242 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being  
1243 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2  
1244 of [SAMLCore].

### 1245 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1246 Participant/Authority**

1247 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or  
1248 participant in a session being terminated. The request is sent using a SAML binding consistent with the  
1249 capability of the responder and the availability of the user agent at the identity provider.

1250 In general, the binding with which the original request was received in step 1 does not dictate the binding  
1251 that may be used in this step except that as noted in step 1, using a synchronous binding that bypasses  
1252 the user agent constrains the identity provider to use a similar binding to propagate additional requests.

1253 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

#### 1254 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity** 1255 **Provider**

1256 The session participant/authority MUST process the <LogoutRequest> message as defined in  
1257 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a  
1258 <LogoutResponse> message containing an appropriate status code to the requesting identity provider  
1259 to complete the SAML protocol exchange.

#### 1260 **Synchronous Bindings (Back-Channel)**

1261 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the  
1262 response is returned directly to complete the synchronous communication. The responder MUST  
1263 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or  
1264 using any other binding-supported mechanism.

#### 1265 **Asynchronous Bindings (Front-Channel)**

1266 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact  
1267 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to  
1268 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be  
1269 used to determine the location of this endpoint and the bindings supported by the identity provider.  
1270 Any asynchronous binding supported by both entities MAY be used.

1271 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is delivered to  
1272 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile  
1273 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity  
1274 to retrieve the <LogoutResponse> message, using for example the SOAP binding.

1275 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or  
1276 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The <LogoutResponse>  
1277 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,  
1278 if used, also provides for an alternate means of authenticating the response issuer when the artifact is  
1279 dereferenced.

1280 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section  
1281 4.4.4.2.

#### 1282 **4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant**

1283 After processing the original session participant's <LogoutRequest> as described in the previous steps  
1284 the identity provider MUST respond to the original request with a <LogoutResponse> containing an  
1285 appropriate status code to complete the SAML protocol exchange.

1286 The response is sent to the original session participant, using a SAML binding consistent with the binding  
1287 used in the original request, the capability of the responder, and the availability of the user agent at the  
1288 identity provider. Assuming an asynchronous binding was used in step 1, then any binding supported by  
1289 both entities MAY be used.

1290 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section  
1291 4.4.4.2.

## 1292 4.4.4 Use of Single Logout Protocol

### 1293 4.4.4.1 <LogoutRequest> Usage

1294 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;  
1295 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-  
1296 format:entity.

1297 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing  
1298 the message or using a binding-specific mechanism.

1299 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in  
1300 the authentication assertion the requester issued or received regarding the session being terminated, per  
1301 the matching rules defined in Section 3.3.4 of [SAMLCore].

1302 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the  
1303 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such  
1304 elements to indicate the termination of all of the principal's applicable sessions.

### 1305 4.4.4.2 <LogoutResponse> Usage

1306 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1307 entity; the Format attribute MUST be omitted or have a value of  
1308 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1309 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1310 the message or using a binding-specific mechanism.

## 1311 4.4.5 Use of Metadata

1312 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported  
1313 bindings and location(s) to which an entity may send requests and responses using this profile.

1314 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>  
1315 element with a use attribute of encryption to determine an appropriate encryption algorithm and  
1316 settings to use, along with a public key to use in delivering a bulk encryption key.

## 1317 4.5 Name Identifier Management Profile

1318 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged  
1319 some form of persistent identifier for a principal with a service provider, allowing them to share a common  
1320 identifier for some length of time. Subsequently, the identity provider may wish to notify the service  
1321 provider of a change in the format and/or value that it will use to identify the same principal in the future.  
1322 Alternatively the service provider may wish to attach its own "alias" for the principal in order to ensure that  
1323 the identity provider will include it when communicating with it in the future about the principal. Finally, one  
1324 of the providers may wish to inform the other that it will no longer issue or accept messages using a  
1325 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management  
1326 protocol is used.

1327 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or  
1328 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A  
1329 front-channel binding may be required, for example, in cases in which direct interaction between the user  
1330 agent and the responding provider is required in order to effect the change.

1331 **4.5.1 Required Information**

1332 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1333 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1334 **Description:** Given below.

1335 **Updates:** None.

1336 **4.5.2 Profile Overview**

1337 Figure 4 illustrates the basic template for the name identifier management profile.

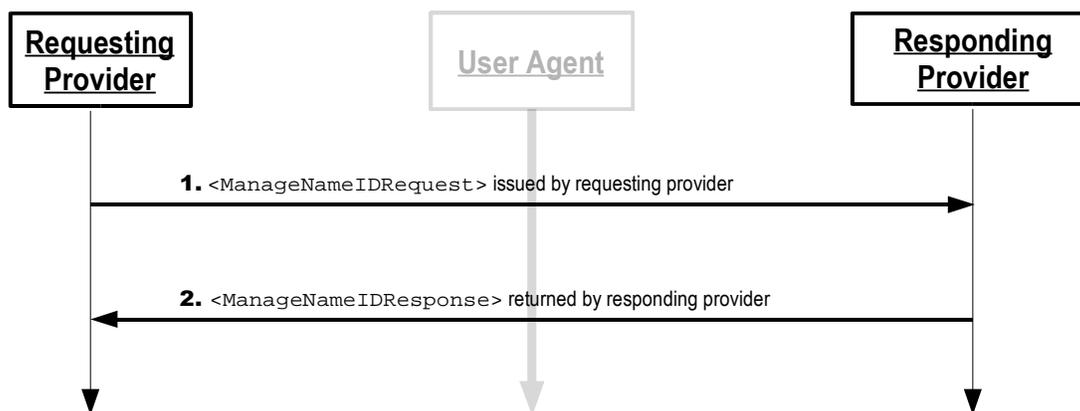


Figure 4

1338 The grayed-out user agent illustrates that the message exchange may pass through the user agent or  
1339 may be a direct exchange between system entities, depending on the SAML binding used to implement  
1340 the profile.

1341 The following steps are described by the profile. Within an individual step, there may be one or more  
1342 actual message exchanges depending on the binding used for that step and other implementation-  
1343 dependent behavior.

1344 **1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1345 In step 1, an identity or service provider initiates the profile by sending a  
1346 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.  
1347 The request may be sent directly to the responding provider or sent indirectly through the user  
1348 agent.

1349 **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1350 In step 2, the responding provider (after processing the request) issues a  
1351 <ManageNameIDResponse> message to the original requesting provider. The response may be  
1352 returned directly to the requesting provider or indirectly through the user agent (if consistent with  
1353 the form of the request in step 1).

1354 **4.5.3 Profile Description**

1355 In the descriptions below, the following is referred to:

## 1356 **Name Identifier Management Service**

1357 This is the name identifier management protocol endpoint at an identity or service provider to which  
1358 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact  
1359 representing them) are delivered. The same or different endpoints MAY be used for requests and  
1360 responses.

### 1361 **4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider**

1362 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another  
1363 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be  
1364 used to determine the location of this endpoint and the bindings supported by the responding provider.

#### 1365 **Synchronous Bindings (Back-Channel)**

1366 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to  
1367 send the request directly to the other provider. The requester MUST authenticate itself to the other  
1368 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported  
1369 mechanism.

#### 1370 **Asynchronous Bindings (Front-Channel)**

1371 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an  
1372 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the  
1373 request to the other provider through the user agent.

1374 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is  
1375 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution  
1376 profile defined in Section 5 is used by the other provider, which makes a callback to the requesting  
1377 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1378 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or  
1379 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The  
1380 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is  
1381 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the  
1382 request issuer when the artifact is dereferenced.

1383 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to  
1384 associate the profile exchange with the original request. The requesting provider SHOULD reveal as  
1385 little information as possible in the RelayState value unless the use of the profile does not require such  
1386 privacy measures.

1387 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section  
1388 4.5.4.1.

### 1389 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1390 Provider**

1391 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After  
1392 processing the message or upon encountering an error, the recipient MUST issue a  
1393 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider  
1394 to complete the SAML protocol exchange.

#### 1395 **Synchronous Bindings (Back-Channel)**

1396 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the  
1397 response is returned directly to complete the synchronous communication. The responder MUST  
1398 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or  
1399 using any other binding-supported mechanism.

#### 1400 **Asynchronous Bindings (Front-Channel)**

1401 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or  
1402 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through  
1403 the user agent to the requesting provider's name identifier management service response endpoint.  
1404 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings  
1405 supported by the requesting provider. Any binding supported by both entities MAY be used.

1406 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is  
1407 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact  
1408 Resolution profile defined in Section 5 is used by the requesting provider, which makes a callback to  
1409 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the  
1410 SOAP binding.

1411 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or  
1412 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The  
1413 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is  
1414 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the  
1415 response issuer when the artifact is dereferenced.

1416 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in  
1417 Section 4.5.4.2.

## 1418 **4.5.4 Use of Name Identifier Management Protocol**

### 1419 **4.5.4.1 <ManageNameIDRequest> Usage**

1420 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;  
1421 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1422 `format:entity`.

1423 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing  
1424 the message or using a binding-specific mechanism.

### 1425 **4.5.4.2 <ManageNameIDResponse> Usage**

1426 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1427 entity; the `Format` attribute MUST be omitted or have a value of  
1428 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1429 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1430 the message or using a binding-specific mechanism.

## 1431 **4.5.5 Use of Metadata**

1432 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported  
1433 bindings and location(s) to which an entity may send requests and responses using this profile.

1434 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>  
1435 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and  
1436 settings to use, along with a public key to use in delivering a bulk encryption key.

1437

## 5 Artifact Resolution Profile

1438 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding  
1439 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML  
1440 protocol messages by reference. This profile describes the use of this protocol with a synchronous  
1441 binding, such as the SOAP binding defined in [SAMLBind].

### 5.1 Required Information

1443 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1444 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1445 **Description:** Given below.

1446 **Updates:** None

### 5.2 Profile Overview

1448 The message exchange and basic processing rules that govern this profile are largely defined by Section  
1449 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to  
1450 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to  
1451 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1452 Figure 5 illustrates the basic template for the artifact resolution profile.

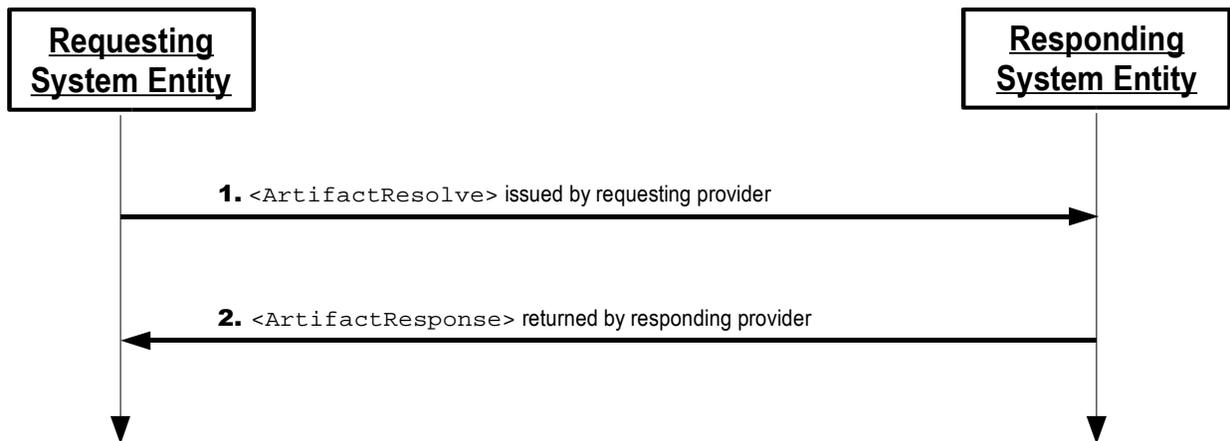


Figure 5

1453 The following steps are described by the profile.

#### 1454 1. <ArtifactResolve> issued by Requesting Entity

1455 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an  
1456 artifact issuer.

## 1457 2. <ArtifactResponse> issued by Responding Entity

1458 In step 2, the responder (after processing the request) issues an <ArtifactResponse>  
1459 message to the requester.

## 1460 5.3 Profile Description

1461 In the descriptions below, the following is referred to:

### 1462 Artifact Resolution Service

1463 This is the artifact resolution protocol endpoint at an artifact issuer to which <ArtifactResolve>  
1464 messages are delivered.

### 1465 5.3.1 <ArtifactResolve> issued by Requesting Entity

1466 To initiate the profile, a requester, having received an artifact and determined the issuer using the  
1467 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact  
1468 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this  
1469 endpoint and the bindings supported by the artifact issuer.

1470 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the  
1471 request directly to the artifact issuer. The requester SHOULD authenticate itself to the responder, either by  
1472 signing the <ArtifactResolve> message or using any other binding-supported mechanism. Specific  
1473 profiles that use the HTTP Artifact binding MAY impose additional requirements such that authentication is  
1474 mandatory.

1475 Profile-specific rules for the contents of the <ArtifactResolve> message are included in Section 5.4.1.

### 1476 5.3.2 <ArtifactResponse> issued by Responding Entity

1477 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After  
1478 processing the message or upon encountering an error, the artifact issuer MUST return an  
1479 <ArtifactResponse> message containing an appropriate status code to the requester to complete the  
1480 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the  
1481 artifact will also be included.

1482 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or  
1483 using any other binding-supported mechanism.

1484 Profile-specific rules for the contents of the <ArtifactResponse> message are included in Section  
1485 5.4.2.

## 1486 5.4 Use of Artifact Resolution Protocol

### 1487 5.4.1 <ArtifactResolve> Usage

1488 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;  
1489 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-  
1490 format:entity.

1491 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by  
1492 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact  
1493 binding MAY impose additional requirements such that authentication is mandatory.

## 1494 **5.4.2 <ArtifactResponse> Usage**

1495 The <Issuer> element MUST be present and MUST contain the unique identifier of the artifact issuer;  
1496 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-  
1497 format:entity.

1498 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1499 the message or using a binding-specific mechanism.

## 1500 **5.5 Use of Metadata**

1501 [SAMLMeta] defines an indexed endpoint element, <md:ArtifactResolutionService>, to describe  
1502 supported bindings and location(s) to which a requester may send requests using this profile. The index  
1503 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's  
1504 EndpointIndex field.

---

## 1505 6 Assertion Query/Request Profile

1506 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis  
1507 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a  
1508 synchronous binding, such as the SOAP binding defined in [SAMLBind].

### 1509 6.1 Required Information

1510 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1511 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1512 **Description:** Given below.

1513 **Updates:** None.

### 1514 6.2 Profile Overview

1515 The message exchange and basic processing rules that govern this profile are largely defined by Section  
1516 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to  
1517 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to  
1518 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1519 Figure 6 illustrates the basic template for the query/request profile.

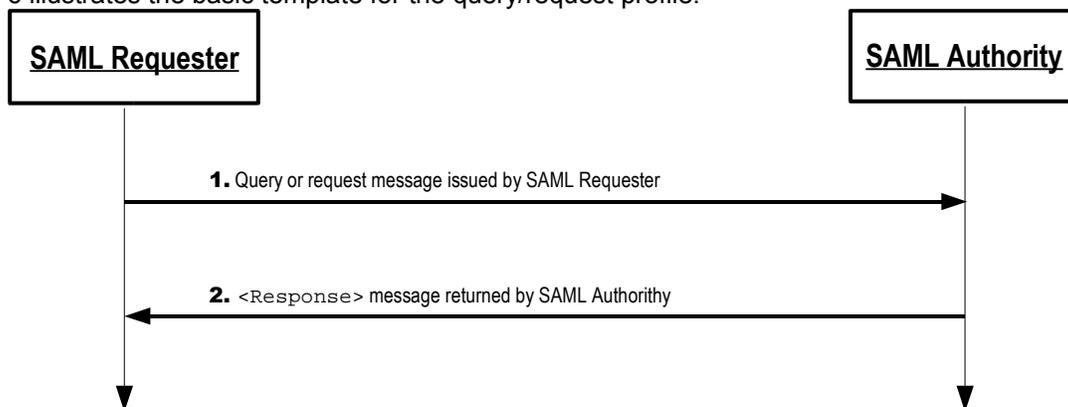


Figure 6

1520 The following steps are described by the profile.

#### 1521 1. Query/Request issued by SAML Requester

1522 In step 1, a SAML requester initiates the profile by sending an `<AssertionIDRequest>`,  
1523 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`  
1524 message to a SAML authority.

#### 1525 2. `<Response>` issued by SAML Authority

1526 In step 2, the responding SAML authority (after processing the query or request) issues a  
1527 `<Response>` message to the SAML requester.

## 1528 **6.3 Profile Description**

1529 In the descriptions below, the following are referred to:

### 1530 **Query/Request Service**

1531 This is the query/request protocol endpoint at a SAML authority to which query or  
1532 `<AssertionIDRequest>` messages are delivered.

### 1533 **6.3.1 Query/Request issued by SAML Requester**

1534 To initiate the profile, a SAML requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`,  
1535 `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's  
1536 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of  
1537 this endpoint and the bindings supported by the SAML authority.

1538 The SAML requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send  
1539 the request directly to the identity provider. The requester SHOULD authenticate itself to the SAML  
1540 authority either by signing the message or using any other binding-supported mechanism.

1541 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

### 1542 **6.3.2 `<Response>` issued by SAML Authority**

1543 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After  
1544 processing the message or upon encountering an error, the SAML authority MUST return a `<Response>`  
1545 message containing an appropriate status code to the SAML requester to complete the SAML protocol  
1546 exchange. If the request is successful in locating one or more matching assertions, they will also be  
1547 included in the response.

1548 The responder SHOULD authenticate itself to the requester, either by signing the `<Response>` or using  
1549 any other binding-supported mechanism.

1550 Profile-specific rules for the contents of the `<Response>` message are included in Section 6.4.2.

## 1551 **6.4 Use of Query/Request Protocol**

### 1552 **6.4.1 Query/Request Usage**

1553 The `<Issuer>` element MUST be present.

1554 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by  
1555 signing the message or using a binding-specific mechanism.

### 1556 **6.4.2 `<Response>` Usage**

1557 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding  
1558 SAML authority; the `Format` attribute MUST be omitted or have a value of  
1559 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this need not necessarily  
1560 match the `<Issuer>` element in the returned assertion(s).

1561 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by  
1562 signing the message or using a binding-specific mechanism.

## 1563 **6.5 Use of Metadata**

1564 [SAMLMeta] defines several endpoint elements, `<md:AssertionIDRequestService>`,  
1565 `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe  
1566 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1567 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can  
1568 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an  
1569 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk  
1570 encryption key.

1571 The various role descriptors MAY contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and  
1572 `<saml:Attribute>` elements (as applicable) to indicate the general ability to support particular name  
1573 identifier formats, attribute profiles, or specific attributes and values. The ability to support any such  
1574 features during a given request is dependent on policy and the discretion of the authority.

1575

## 7 Name Identifier Mapping Profile

1576 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a  
1577 different name identifier for the same principal. This profile describes the use of this protocol with a  
1578 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for  
1579 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

### 7.1 Required Information

1581 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1582 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1583 **Description:** Given below.

1584 **Updates:** None.

### 7.2 Profile Overview

1586 The message exchange and basic processing rules that govern this profile are largely defined by Section  
1587 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to  
1588 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to  
1589 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1590 Figure 7 illustrates the basic template for the name identifier mapping profile.



Figure 7

1591 The following steps are described by the profile.

#### 1592 1. <NameIDMappingRequest> issued by Requesting Entity

1593 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to  
1594 an identity provider.

#### 1595 2. <NameIDMappingResponse> issued by Identity Provider

1596 In step 2, the responding identity provider (after processing the request) issues a  
1597 <NameIDMappingResponse> message to the requester.

## 1598 **7.3 Profile Description**

1599 In the descriptions below, the following is referred to:

### 1600 **Name Identifier Mapping Service**

1601 This is the name identifier mapping protocol endpoint at an identity provider to which  
1602 <NameIDMappingRequest> messages are delivered.

#### 1603 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1604 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's  
1605 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the  
1606 location of this endpoint and the bindings supported by the identity provider.

1607 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the  
1608 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,  
1609 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1610 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in  
1611 Section 7.4.1.

#### 1612 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1613 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].  
1614 After processing the message or upon encountering an error, the identity provider MUST return a  
1615 <NameIDMappingResponse> message containing an appropriate status code to the requester to  
1616 complete the SAML protocol exchange.

1617 The responder MUST authenticate itself to the requester, either by signing the  
1618 <NameIDMappingResponse> or using any other binding-supported mechanism.

1619 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in  
1620 Section 7.4.2.

## 1621 **7.4 Use of Name Identifier Mapping Protocol**

### 1622 **7.4.1 <NameIDMappingRequest> Usage**

1623 The <Issuer> element MUST be present.

1624 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing  
1625 the message or using a binding-specific mechanism.

### 1626 **7.4.2 <NameIDMappingResponse> Usage**

1627 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1628 identity provider; the Format attribute MUST be omitted or have a value of  
1629 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1630 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1631 the message or using a binding-specific mechanism.

1632 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In  
1633 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester  
1634 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it  
1635 in subsequent protocol messages or assertions.

#### 1636 **7.4.2.1 Limiting Use of Mapped Identifier**

1637 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning  
1638 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but  
1639 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>  
1640 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>  
1641 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying  
1642 parties, and MUST be signed for integrity protection.

### 1643 **7.5 Use of Metadata**

1644 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported  
1645 bindings and location(s) to which a requester may send requests using this profile.

1646 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's  
1647 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate  
1648 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

---

## 1649 8 SAML Attribute Profiles

### 1650 8.1 Basic Attribute Profile

1651 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with  
1652 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas  
1653 to validate syntax.

#### 1654 8.1.1 Required Information

1655 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1656 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1657 **Description:** Given below.

1658 **Updates:** None.

#### 1659 8.1.2 SAML Attribute Naming

1660 The NameFormat XML attribute in <Attribute> elements MUST be  
1661 urn:oasis:names:tc:SAML:2.0:attrname-format:basic.

1662 The Name XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

##### 1663 8.1.2.1 Attribute Name Comparison

1664 Two <Attribute> elements refer to the same SAML attribute if and only if the values of their Name XML  
1665 attributes are equal in the sense of Section 3.3.6 of [Schema2].

#### 1666 8.1.3 Profile-Specific XML Attributes

1667 No additional XML attributes are defined for use with the <Attribute> element.

#### 1668 8.1.4 SAML Attribute Values

1669 The schema type of the contents of the <AttributeValue> element MUST be drawn from one of the  
1670 types defined in Section 3.3 of [Schema2]. The xsi:type attribute MUST be present and be given the  
1671 appropriate value.

#### 1672 8.1.5 Example

```
1673 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1674     Name="FirstName">  
1675     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1676 </saml:Attribute>
```

## 1677 8.2 X.500/LDAP Attribute Profile

1678 Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory  
1679 Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model  
1680 information to be stored in these directories. In particular, in X.500, attribute type definitions are used to  
1681 specify the syntax and other features of attributes, the basic information storage unit in a directory (this

1682 document refers to these as “directory attributes”). Directory attribute types are defined in schema in the  
1683 X.500 and LDAP specifications themselves, schema in other public documents (such as the  
1684 Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and  
1685 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of  
1686 these directory attribute types in the context of SAML attribute statements, without having to manually  
1687 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.  
1688 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such  
1689 attributes when expressed as SAML attributes.

## 1690 8.2.1 Required Information

1691 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace  
1692 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

1693 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1694 **Description:** Given below.

1695 **Updates:** None.

## 1696 8.2.2 SAML Attribute Naming

1697 The `NameFormat` XML attribute in `<Attribute>` elements MUST be  
1698 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1699 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.  
1700 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory  
1701 attribute type.

1702 Example:

```
1703 urn:oid:2.5.4.3
```

1704 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,  
1705 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1706 For purposes of human readability, there may also be a requirement for some applications to carry an  
1707 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in  
1708 [SAMLCore]) MAY be used for this purpose. If the definition of the directory attribute type includes one or  
1709 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, SHOULD be  
1710 one of the defined descriptors.

### 1711 8.2.2.1 Attribute Name Comparison

1712 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1713 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the  
1714 comparison.

## 1715 8.2.3 Profile-Specific XML Attributes

1716 No additional XML attributes are defined for use with the `<Attribute>` element.

## 1717 8.2.4 SAML Attribute Values

1718 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute  
1719 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax  
1720 which specifies how attribute or assertion values conforming to the syntax are to be represented when  
1721 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding

1722 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of  
1723 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to  
1724 this profile may define attribute value formats for directory attributes whose syntaxes specify other  
1725 encodings.

1726 To represent the encoding rules in use for a particular attribute value, the <AttributeValue> element  
1727 MUST contain an XML attribute named `Encoding` defined in the XML namespace  
1728 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`.

1729 For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8  
1730 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the  
1731 content of the <AttributeValue> element, with no additional whitespace. In such cases, the  
1732 `xsi:type` XML attribute MUST be set to **xs:string**. The profile-specific `Encoding` XML attribute is  
1733 provided, with a value of `LDAP`.

1734 A list of some LDAP attribute syntaxes to which this applies is:

1735	Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1736	Bit String	1.3.6.1.4.1.1466.115.121.1.6
1737	Boolean	1.3.6.1.4.1.1466.115.121.1.7
1738	Country String	1.3.6.1.4.1.1466.115.121.1.11
1739	DN	1.3.6.1.4.1.1466.115.121.1.12
1740	Directory String	1.3.6.1.4.1.1466.115.121.1.15
1741	Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1742	Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1743	IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1744	INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1745	LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1746	Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1747	Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1748	Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1749	Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1750	Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1751	Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1752	Octet String	1.3.6.1.4.1.1466.115.121.1.40
1753	OID	1.3.6.1.4.1.1466.115.121.1.38
1754	Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1755	Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1756	Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1757	Printable String	1.3.6.1.4.1.1466.115.121.1.44
1758	Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1759	Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1760	UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1761 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>  
1762 element, by base64-encoding [RFC2045] the encompassing ASN.1 OCTET STRING-encoded LDAP  
1763 attribute value. The `xsi:type` XML attribute MUST be set to **xs:base64Binary**. The profile-specific  
1764 `Encoding` XML attribute is provided, with a value of `"LDAP"`.

1765 When comparing SAML attribute values for equality, the matching rules specified for the corresponding  
1766 directory attribute type MUST be observed (case sensitivity, for example).

## 1767 8.2.5 Profile-Specific Schema

1768 The following schema listing shows how the profile-specific `Encoding` XML attribute is defined  
1769 [SAMLX500-xsd]:

```

1770 <schema
1771   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1772   xmlns="http://www.w3.org/2001/XMLSchema"
1773   elementFormDefault="unqualified"
1774   attributeFormDefault="unqualified"
1775   blockDefault="substitution"
1776   version="2.0">
1777   <annotation>
1778     <documentation>
1779       Document identifier: saml-schema-x500-2.0
1780       Location: http://docs.oasis-open.org/security/saml/v2.0/
1781       Revision history:
1782         V2.0 (March, 2005):
1783         Custom schema for X.500 attribute profile, first published in
1784 SAML 2.0.
1785     </documentation>
1786   </annotation>
1787   <attribute name="Encoding" type="string"/>
1788 </schema>

```

## 1789 8.2.6 Example

1790 The following is an example of a mapping of the "givenName" directory attribute, representing the SAML  
1791 assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory  
1792 String.

```

1793 <saml:Attribute
1794   xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1795     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1796     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
1797   <saml:AttributeValue xsi:type="xs:string"
1798     x500:Encoding="LDAP">Steven</saml:AttributeValue>
1799 </saml:Attribute>

```

## 1800 8.3 UUID Attribute Profile

1801 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and  
1802 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with  
1803 a UUID.

### 1804 8.3.1 Required Information

1805 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1806 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1807 **Description:** Given below.

1808 **Updates:** None.

### 1809 8.3.2 UUID and GUID Background

1810 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to  
1811 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs  
1812 were originally used in the Network Computing System (NCS), and then used in the Open Software  
1813 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in  
1814 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1815 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of  
1816 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,  
1817 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1818 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1819 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a  
1820 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

### 1821 **8.3.3 SAML Attribute Naming**

1822 The `NameFormat` XML attribute in `<Attribute>` elements MUST be  
1823 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1824 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace  
1825 described in [Mealling] is used. In this approach the `Name` XML attribute is based on the URN form of the  
1826 underlying UUID that identifies the attribute.

1827 Example:

1828 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1829 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used  
1830 in the `Name` XML attribute.

1831 For purposes of human readability, there may also be a requirement for some applications to carry an  
1832 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in  
1833 [SAMLCORE]) MAY be used for this purpose.

#### 1834 **8.3.3.1 Attribute Name Comparison**

1835 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1836 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt]. The  
1837 `FriendlyName` attribute plays no role in the comparison.

### 1838 **8.3.4 Profile-Specific XML Attributes**

1839 No additional XML attributes are defined for use with the `<Attribute>` element.

### 1840 **8.3.5 SAML Attribute Values**

1841 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be  
1842 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST  
1843 be set to **xs:anyURI**.

1844 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`  
1845 element.

### 1846 **8.3.6 Example**

1847 The following is an example of a DCE Extended Registry Attribute, the "pre\_auth\_req" setting, which has a  
1848 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1849 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1850     Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"  
1851     FriendlyName="pre_auth_req">  
1852     <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>  
1853 </saml:Attribute>
```

## 1854 8.4 DCE PAC Attribute Profile

1855 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names  
1856 and values. It is used to standardize a mapping between the primary information that makes up a DCE  
1857 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in  
1858 Section 8.3.

### 1859 8.4.1 Required Information

1860 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace  
1861 assigned in the corresponding DCE PAC attribute profile schema document [SAML DCE-xsd])

1862 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1863 **Description:** Given below.

1864 **Updates:** None.

### 1865 8.4.2 PAC Description

1866 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of  
1867 information is common across principals and makes up the bulk of a DCE identity:

- 1868 • The principal's DCE "realm" or "cell"
- 1869 • The principal's unique identifier
- 1870 • The principal's primary DCE local group membership
- 1871 • The principal's set of DCE local group memberships (multi-valued)
- 1872 • The principal's set of DCE foreign group memberships (multi-valued)

1873 The primary value(s) of each of these attributes is a UUID.

### 1874 8.4.3 SAML Attribute Naming

1875 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual  
1876 specific attribute names, rather than a naming convention.

1877 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST  
1878 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1879 For purposes of human readability, there may also be a requirement for some applications to carry an  
1880 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in  
1881 [SAMLCore]) MAY be used for this purpose.

1882 See Section 8.4.6 for the specific attribute names defined by this profile.

#### 1883 8.4.3.1 Attribute Name Comparison

1884 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1885 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt]. The  
1886 `FriendlyName` attribute plays no role in the comparison.

### 1887 8.4.4 Profile-Specific XML Attributes

1888 No additional XML attributes are defined for use with the `<Attribute>` element.

## 1889 8.4.5 SAML Attribute Values

1890 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described  
1891 in Section 8.3.5 of the UUID profile is used to represent such values.

1892 However, additional information associated with the UUID value is permitted by this profile, consisting of a  
1893 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional  
1894 information is carried in the <AttributeValue> element in FriendlyName and Realm XML attributes  
1895 defined in the XML namespace urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE. Note  
1896 that this is not the same as the FriendlyName XML attribute defined in [SAMLCore], although it has the  
1897 same basic purpose.

1898 The following schema listing shows how the profile-specific XML attributes and complex type used in an  
1899 xsi:type specification are defined [SAML DCE-xsd]:

```
1900 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1901   xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1902   xmlns="http://www.w3.org/2001/XMLSchema"  
1903   elementFormDefault="unqualified"  
1904   attributeFormDefault="unqualified"  
1905   blockDefault="substitution"  
1906   version="2.0">  
1907   <annotation>  
1908     <documentation>  
1909       Document identifier: saml-schema-dce-2.0  
1910       Location: http://docs.oasis-open.org/security/saml/v2.0/  
1911       Revision history:  
1912       V2.0 (March, 2005):  
1913         Custom schema for DCE attribute profile, first published in  
1914 SAML 2.0.  
1915     </documentation>  
1916   </annotation>  
1917   <complexType name="DCEValueType">  
1918     <simpleContent>  
1919       <extension base="anyURI">  
1920         <attribute ref="dce:Realm" use="optional"/>  
1921         <attribute ref="dce:FriendlyName" use="optional"/>  
1922       </extension>  
1923     </simpleContent>  
1924   </complexType>  
1925   <attribute name="Realm" type="anyURI"/>  
1926   <attribute name="FriendlyName" type="string"/>  
1927 </schema>
```

## 1928 8.4.6 Attribute Definitions

1929 The following are the set of SAML attributes defined by this profile. In each case, an xsi:type XML  
1930 attribute MAY be included in the <AttributeValue> element, but MUST have the value  
1931 **dce:DCEValueType**, where the dce prefix is arbitrary and MUST be bound to the XML namespace  
1932 urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE.

1933 Note that such use of xsi:type will require validating attribute consumers to include the extension  
1934 schema defined by this profile.

### 1935 8.4.6.1 Realm

1936 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

1937 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

1938 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion

1939 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the  
1940 realm's string name.

#### 1941 **8.4.6.2 Principal**

1942 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

1943 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

1944 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion  
1945 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing  
1946 the principal's string name.

1947 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form  
1948 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section  
1949 8.4.6.1).

#### 1950 **8.4.6.3 Primary Group**

1951 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

1952 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

1953 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion  
1954 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing  
1955 the group's string name.

1956 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form  
1957 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section  
1958 8.4.6.1).

#### 1959 **8.4.6.4 Groups**

1960 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

1961 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

1962 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE group membership  
1963 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing  
1964 the group's string name.

1965 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form  
1966 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section  
1967 8.4.6.1).

#### 1968 **8.4.6.5 Foreign Groups**

1969 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

1970 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

1971 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE foreign group  
1972 membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute  
1973 containing the group's string name.

1974 The profile-specific `Realm` XML attribute MUST be included and MUST contain a UUID in URN form  
1975 identifying the DCE realm/cell of the foreign group.

## 1976 8.4.7 Example

1977 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE  
1978 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local  
1979 groups and an "engineers" foreign group.

```
1980 <saml:Assertion xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1981 ...>  
1982 <saml:Issuer>...</saml:Issuer>  
1983 <saml:Subject>...</saml:Subject>  
1984 <saml:AttributeStatement>  
1985 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1986 Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">  
1987 <saml:AttributeValue xsi:type="dce:DCEValueType"  
1988 dce:FriendlyName="example.com">  
1989 urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b  
1990 </saml:AttributeValue>  
1991 </saml:Attribute>  
1992 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1993 Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">  
1994 <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">  
1995 urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b  
1996 </saml:AttributeValue>  
1997 </saml:Attribute>  
1998 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1999 Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">  
2000 <saml:AttributeValue xsi:type="dce:DCEValueType"  
2001 dce:FriendlyName="cubicle-dwellers">  
2002 urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2003 </saml:AttributeValue>  
2004 </saml:Attribute>  
2005 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2006 Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">  
2007 <saml:AttributeValue xsi:type="dce:DCEValueType"  
2008 dce:FriendlyName="cubicle-dwellers">  
2009 urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2010 </saml:AttributeValue>  
2011 <saml:AttributeValue xsi:type="dce:DCEValueType"  
2012 dce:FriendlyName="underpaid">  
2013 urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b  
2014 </saml:AttributeValue>  
2015 </saml:Attribute>  
2016 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2017 Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-  
2018 groups">  
2019 <saml:AttributeValue xsi:type="dce:DCEValueType"  
2020 dce:FriendlyName="engineers"  
2021 dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">  
2022 urn:uuid:00099cfl-a355-10f9-9e95-004005b13a2b  
2023 </saml:AttributeValue>  
2024 </saml:Attribute>  
2025 </saml:AttributeStatement>  
2026 </saml:Assertion>
```

## 2027 8.5 XACML Attribute Profile

2028 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS  
2029 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute  
2030 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML  
2031 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute  
2032 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into  
2033 XACML attributes and used as input to XACML authorization decisions.

## 2034 8.5.1 Required Information

2035 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace  
2036 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

2037 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

2038 **Description:** Given below.

2039 **Updates:** None.

## 2040 8.5.2 SAML Attribute Naming

2041 The `NameFormat` XML attribute in `<Attribute>` elements MUST be

2042 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

2043 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

2044 For purposes of human readability, there may also be a requirement for some applications to carry an  
2045 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in  
2046 [SAMLCore]) MAY be used for this purpose, but is not translatable into an XACML attribute equivalent.

### 2047 8.5.2.1 Attribute Name Comparison

2048 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
2049 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

## 2050 8.5.3 Profile-Specific XML Attributes

2051 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-  
2052 valued XML attribute called `DataType` is defined in the XML namespace

2053 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

2054 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified  
2055 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

2056 While in principle any URI reference can be used as a data type, the standard values to be used are  
2057 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then  
2058 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values  
2059 must be extended to support the new data types.

## 2060 8.5.4 SAML Attribute Values

2061 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed  
2062 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data  
2063 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute  
2064 SHOULD also be used on the `<AttributeValue>` element(s).

## 2065 8.5.5 Profile-Specific Schema

2066 The following schema listing shows how the profile-specific `DataType` XML attribute is defined  
2067 [SAMLXAC-xsd]:

```
2068 <schema  
2069   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
2070   xmlns="http://www.w3.org/2001/XMLSchema"  
2071   elementFormDefault="unqualified"
```

```

2072     attributeFormDefault="unqualified"
2073     blockDefault="substitution"
2074     version="2.0">
2075     <annotation>
2076         <documentation>
2077             Document identifier: saml-schema-xacml-2.0
2078             Location: http://docs.oasis-open.org/security/saml/v2.0/
2079             Revision history:
2080             V2.0 (March, 2005):
2081                 Custom schema for XACML attribute profile, first published in
2082 SAML 2.0.
2083         </documentation>
2084     </annotation>
2085     <attribute name="DataType" type="anyURI"/>
2086 </schema>

```

## 2087 **8.5.6 Example**

2088 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the  
2089 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple  
2090 attribute profiles when they are compatible with each other.

```

2091 <saml:Attribute
2092 xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2093     xmlns:ldaprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2094     xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2095     ldaprof:Encoding="LDAP"
2096     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2097     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2098     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>
2099 </saml:Attribute>

```

---

## 9 References

2100

- 2101 **[AES]** FIPS-197, Advanced Encryption Standard (AES). See <http://www.nist.gov/>.
- 2102 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”.  
2103 See <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2104 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic  
2105 notation, ITU-T Recommendation X.680, July 2002. See  
2106 [http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680)  
2107 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2108 **[eduPerson]** eduPerson.Idif. See <http://www.educause.edu/eduperson>.
- 2109 **[LDAP]** J. Hodges et al. *Lightweight Directory Access Protocol (v3): Technical Specification*.  
2110 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2111 **[Mealling]** P Leach et al. *A UUID URN Namespace*. IETF Internet-Draft, December 2004. See  
2112 <http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt>.
- 2113 **[MSURL]** Microsoft technical support article. See  
2114 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2115 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See  
2116 [http://wp.netscape.com/newsref/std/cookie\\_spec.html](http://wp.netscape.com/newsref/std/cookie_spec.html).
- 2117 **[PAOS]** R. Aarts. *Liberty Reverse HTTP Binding for SOAP Specification Version 1.0*. Liberty  
2118 Alliance Project, 2003. See <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>.
- 2119 **[Rescorla-Sec]** E. Rescorla et al. *Guidelines for Writing RFC Text on Security Considerations*. IETF  
2120 RFC 3552, July 2003. See <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2121 **[RFC1738]** T. Berners-Lee et al. *Uniform Resource Locators (URL)*. IETF RFC 1738, December  
2122 1994. See <http://www.ietf.org/rfc/rfc1738.txt>.
- 2123 **[RFC1750]** D. Eastlake et al. *Randomness Recommendations for Security*. IETF RFC 1750,  
2124 December 1994. See <http://www.ietf.org/rfc/rfc1750.txt>.
- 2125 **[RFC1945]** T. Berners-Lee et al. *Hypertext Transfer Protocol – HTTP/1.0*. IETF RFC 1945, May  
2126 1996. See <http://www.ietf.org/rfc/rfc1945.txt>.
- 2127 **[RFC2045]** N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of*  
2128 *Internet Message Bodies*. IETF RFC 2045, November 1996. See  
2129 <http://www.ietf.org/rfc/rfc2045.txt>.
- 2130 **[RFC2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC  
2131 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 2132 **[RFC2246]** T. Dierks. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See  
2133 <http://www.ietf.org/rfc/rfc2246.txt>.
- 2134 **[RFC2256]** M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*. IETF RFC  
2135 2256, December 1997. See <http://www.ietf.org/rfc/rfc2256.txt>.
- 2136 **[RFC2279]** F. Yergeau. *UTF-8, a transformation format of ISO 10646*. IETF RFC 2279, January  
2137 1998. See <http://www.ietf.org/rfc/rfc2279.txt>.
- 2138 **[RFC2616]** R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999.  
2139 See <http://www.ietf.org/rfc/rfc2616.txt>.
- 2140 **[RFC2617]** J. Franks et al. *HTTP Authentication: Basic and Digest Access Authentication*. IETF  
2141 RFC 2617, June 1999. See <http://www.ietf.org/rfc/rfc2617.txt>.
- 2142 **[RFC2798]** M. Smith. *Definition of the inetOrgPerson LDAP Object Class*. IETF RFC 2798, April  
2143 2000. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2144 **[RFC2965]** D. Cristol et al. *HTTP State Management Mechanism*. IETF RFC 2965, October 2000.  
2145 See <http://www.ietf.org/rfc/rfc2965.txt>.

2146	<b>[RFC3061]</b>	M. Mealling. <i>A URN Namespace of Object Identifiers</i> . IETF RFC 3061, February 2001. See <a href="http://www.ietf.org/rfc/rfc3061.txt">http://www.ietf.org/rfc/rfc3061.txt</a> .
2147		
2148	<b>[SAMLBind]</b>	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2149		
2150		
2151	<b>[SAMLConform]</b>	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2152		
2153		
2154	<b>[SAMLCore]</b>	S. Cantor et al. <i>Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-core-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2155		
2156		
2157	<b>[SAML DCE-xsd]</b>	S. Cantor et al. SAML DCE PAC attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-dce-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2158		
2159		
2160	<b>[SAML ECP-xsd]</b>	S. Cantor et al. SAML ECP profile schema. OASIS SSTC, March 2005. Document ID saml-schema-ecp-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2161		
2162	<b>[SAML Gloss]</b>	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2163		
2164		
2165	<b>[SAML X500-xsd]</b>	S. Cantor et al. SAML X.500/LDAP attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-x500-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2166		
2167		
2168	<b>[SAML Meta]</b>	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2169		
2170		
2171	<b>[SAML Reqs]</b>	Darren Platt et al. <i>OASIS Security Services Use Cases and Requirements</i> . OASIS SSTC, May 2001. Document ID draft-sstc-saml-reqs-01. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2172		
2173		
2174	<b>[SAML Sec]</b>	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2175		
2176		
2177	<b>[SAML Web]</b>	OASIS Security Services Technical Committee website, <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2178		
2179	<b>[SAML XAC-xsd]</b>	S. Cantor et al. SAML XACML attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-xacml-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2180		
2181		
2182	<b>[Schema1]</b>	H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> . Note that this specification normatively references [Schema2], listed below.
2183		
2184		
2185	<b>[Schema2]</b>	Paul V. Biron, Ashok Malhotra. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. See <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a> .
2186		
2187	<b>[SESSION]</b>	RL 'Bob' Morgan. <i>Support of target web server sessions in Shibboleth</i> . Shibboleth, May 2001. See <a href="http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt">http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt</a> .
2188		
2189		
2190	<b>[ShibMarlena]</b>	Marlena Erdos et al. <i>Shibboleth Architecture DRAFT v05</i> . Shibboleth, May 2002. See <a href="http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html">http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html</a> .
2191		
2192	<b>[SOAP1.1]</b>	D. Box et al. <i>Simple Object Access Protocol (SOAP) 1.1</i> . World Wide Web Consortium Note, May 2000. See <a href="http://www.w3.org/TR/SOAP">http://www.w3.org/TR/SOAP</a> .
2193		
2194	<b>[SSL3]</b>	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
2195	<b>[WEBSSO]</b>	RL 'Bob' Morgan. <i>Interactions between Shibboleth and local-site web sign-on services</i> . Shibboleth, April 2001. See <a href="http://middleware.internet2.edu/shibboleth/docs/draft-">http://middleware.internet2.edu/shibboleth/docs/draft-</a>
2196		

2197		<a href="#">morgan-shibboleth-websso-00.txt</a> .
2198	<b>[X.500]</b>	Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services. ITU-T Recommendation X.500, February 2001. See
2199		<a href="http://www.itu.int/rec/recommendation.asp?type=folders&amp;lang=e&amp;parent=T-REC-X.500">http://www.itu.int/rec/recommendation.asp?type=folders&amp;lang=e&amp;parent=T-REC-</a>
2200		<a href="http://www.itu.int/rec/recommendation.asp?type=folders&amp;lang=e&amp;parent=T-REC-X.500">X.500</a> .
2201		
2202	<b>[XMLEnc]</b>	D. Eastlake et al. <i>XML Encryption Syntax and Processing</i> . World Wide Web Consortium Recommendation, December 2002. See
2203		<a href="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/">http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/</a> .
2204		
2205	<b>[XMLSig]</b>	D. Eastlake et al. <i>XML-Signature Syntax and Processing</i> . World Wide Web Consortium Recommendation, February 2002. See <a href="http://www.w3.org/TR/xmlsig-core/">http://www.w3.org/TR/xmlsig-</a>
2206		<a href="http://www.w3.org/TR/xmlsig-core/">core/</a> .
2207		
2208	<b>[XACML]</b>	T. Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Versions 1.0, 1.1, and 2.0. Available on the OASIS XACML TC web page at <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml">http://www.oasis-</a>
2209		<a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml">open.org/committees/tc_home.php?wg_abbrev=xacml</a> .
2210		

---

## Appendix A. Acknowledgments

2212 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
2213 Committee, whose voting members at the time of publication were:

- 2214 • Conor Cahill, AOL
- 2215 • John Hughes, Atos Origin
- 2216 • Hal Lockhart, BEA Systems
- 2217 • Mike Beach, Boeing
- 2218 • Rebekah Metz, Booz Allen Hamilton
- 2219 • Rick Randall, Booz Allen Hamilton
- 2220 • Ronald Jacobson, Computer Associates
- 2221 • Gavenraj Sodhi, Computer Associates
- 2222 • Thomas Wisniewski, Entrust
- 2223 • Carolina Canales-Valenzuela, Ericsson
- 2224 • Dana Kaufman, Forum Systems
- 2225 • Irving Reid, Hewlett-Packard
- 2226 • Guy Denton, IBM
- 2227 • Heather Hinton, IBM
- 2228 • Maryann Hondo, IBM
- 2229 • Michael McIntosh, IBM
- 2230 • Anthony Nadalin, IBM
- 2231 • Nick Ragouzis, Individual
- 2232 • Scott Cantor, Internet2
- 2233 • Bob Morgan, Internet2
- 2234 • Peter Davis, Neustar
- 2235 • Jeff Hodges, Neustar
- 2236 • Frederick Hirsch, Nokia
- 2237 • Senthil Sengodan, Nokia
- 2238 • Abbie Barbir, Nortel Networks
- 2239 • Scott Kiestler, Novell
- 2240 • Cameron Morris, Novell
- 2241 • Paul Madsen, NTT
- 2242 • Steve Anderson, OpenNetwork
- 2243 • Ari Kermaier, Oracle
- 2244 • Vamsi Motukuru, Oracle
- 2245 • Darren Platt, Ping Identity
- 2246 • Prateek Mishra, Principal Identity
- 2247 • Jim Lien, RSA Security
- 2248 • John Linn, RSA Security
- 2249 • Rob Philpott, RSA Security
- 2250 • Dipak Chopra, SAP
- 2251 • Jahan Moreh, Sigaba
- 2252 • Bhavna Bhatnagar, Sun Microsystems
- 2253 • Eve Maler, Sun Microsystems

- 2254 • Ronald Monzillo, Sun Microsystems
- 2255 • Emily Xu, Sun Microsystems
- 2256 • Greg Whitehead, Trustgenix

2257 The editors also would like to acknowledge the following former SSTC members for their contributions to  
2258 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 2259 • Stephen Farrell, Baltimore Technologies
- 2260 • David Orchard, BEA Systems
- 2261 • Krishna Sankar, Cisco Systems
- 2262 • Zahid Ahmed, CommerceOne
- 2263 • Tim Alsop, CyberSafe Limited
- 2264 • Carlisle Adams, Entrust
- 2265 • Tim Moses, Entrust
- 2266 • Nigel Edwards, Hewlett-Packard
- 2267 • Joe Pato, Hewlett-Packard
- 2268 • Bob Blakley, IBM
- 2269 • Marlena Erdos, IBM
- 2270 • Marc Chanliau, Netegrity
- 2271 • Chris McLaren, Netegrity
- 2272 • Lynne Rosenthal, NIST
- 2273 • Mark Skall, NIST
- 2274 • Charles Knouse, Oblix
- 2275 • Simon Godik, Overxeer
- 2276 • Charles Norwood, SAIC
- 2277 • Evan Prodromou, Securant
- 2278 • Robert Griffin, RSA Security (former editor)
- 2279 • Sai Allarvarpu, Sun Microsystems
- 2280 • Gary Ellison, Sun Microsystems
- 2281 • Chris Ferris, Sun Microsystems
- 2282 • Mike Myers, Traceroute Security
- 2283 • Phillip Hallam-Baker, VeriSign (former editor)
- 2284 • James Vanderbeek, Vodafone
- 2285 • Mark O'Neill, Vordel
- 2286 • Tony Palmer, Vordel

2287 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
2288 input to the OASIS Security Assertions Markup Language specifications:

- 2289 • Thomas Gross, IBM
- 2290 • Birgit Pfitzmann, IBM

---

2291 **Appendix B. Notices**

2292 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
2293 might be claimed to pertain to the implementation or use of the technology described in this document or  
2294 the extent to which any license under such rights might or might not be available; neither does it represent  
2295 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
2296 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
2297 available for publication and any assurances of licenses to be made available, or the result of an attempt  
2298 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
2299 users of this specification, can be obtained from the OASIS Executive Director.

2300 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
2301 other proprietary rights which may cover technology that may be required to implement this specification.  
2302 Please address the information to the OASIS Executive Director.

2303 **Copyright © OASIS Open 2005. All Rights Reserved.**

2304 This document and translations of it may be copied and furnished to others, and derivative works that  
2305 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
2306 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
2307 this paragraph are included on all such copies and derivative works. However, this document itself may  
2308 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
2309 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
2310 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
2311 into languages other than English.

2312 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
2313 or assigns.

2314 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2315 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
2316 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
2317 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.