



---

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

Committee Draft 01, 18 August 2004

**Document identifier:**

sstc-saml-profiles-2.0-cd-01

**Location:**

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

**Editors:**

John Hughes, Entegriety Solutions  
Scott Cantor, Internet2  
Prateek Mishra, Netegrity  
Frederick Hirsch, Nokia  
Rob Philpott, RSA Security  
Jeff Hodges, Sun Microsystems  
Eve Maler, Sun Microsystems

**SAML V2.0 Contributors:**

Conor P. Cahill, AOL  
Hal Lockhart, BEA Systems  
Michael Beach, Boeing  
Rick Randall, Booze, Allen, Hamilton  
Tim Alsop, Cybersafe  
Nick Ragouzis, Enosis  
John Hughes, Entegriety Solutions  
Paul Madsen, Entrust  
Irving Reid, Hewlett-Packard  
Paula Austel, IBM  
Maryann Hondo, IBM  
Michael McIntosh, IBM  
Tony Nadalin, IBM  
Scott Cantor, Internet2  
RL 'Bob' Morgan, Internet2  
Rebekah Metz, NASA  
Prateek Mishra, Netegrity  
Peter C Davis, Neustar  
Frederick Hirsch, Nokia  
John Kemp, Nokia  
Charles Knouse, Oblix  
Steve Anderson, OpenNetwork  
John Linn, RSA Security  
Rob Philpott, RSA Security  
Jahan Moreh, Sigaba  
Anne Anderson, Sun Microsystems

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

49 **Abstract:**

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

53 **Status:**

54 This is a **Committee Draft** approved by the Security Services Technical Committee on 17 August  
55 2004.

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

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

---

## Table of Contents

65		
66	1 Introduction.....	7
67	1.1 Profile Concepts.....	7
68	1.2 Notation.....	7
69	2 Specification of Additional Profiles.....	9
70	2.1 Guidelines for Specifying Profiles.....	9
71	2.2 Guidelines for Specifying Attribute Profiles.....	9
72	3 Confirmation Method Identifiers.....	11
73	3.1 Holder of Key.....	11
74	3.2 Sender Vouches.....	11
75	3.3 Bearer.....	12
76	4 SSO Profiles of SAML.....	13
77	4.1 Web Browser SSO Profile.....	13
78	4.1.1 Required Information.....	13
79	4.1.2 Profile Overview.....	13
80	4.1.3 Profile Description.....	15
81	4.1.3.1 HTTP Request to Service Provider.....	15
82	4.1.3.2 Service Provider Determines Identity Provider.....	15
83	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	15
84	4.1.3.4 Identity Provider Identifies Principal.....	16
85	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	16
86	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	16
87	4.1.4 Use of Authentication Request Protocol.....	16
88	4.1.4.1 <AuthnRequest> Usage.....	17
89	4.1.4.2 <Response> Usage.....	17
90	4.1.4.3 <Response> Message Processing Rules.....	18
91	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	18
92	4.1.4.5 POST-Specific Processing Rules.....	18
93	4.1.5 Unsolicited Responses.....	18
94	4.1.6 Use of Metadata.....	19
95	4.2 Enhanced Client or Proxy (ECP) Profile.....	19
96	4.2.1 Required Information.....	20
97	4.2.2 Profile Overview.....	20
98	4.2.3 Profile Description.....	22
99	4.2.3.1 ECP issues HTTP Request to Service Provider.....	22
100	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	22
101	4.2.3.3 ECP Determines Identity Provider.....	23
102	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	23
103	4.2.3.5 Identity Provider Identifies Principal.....	23
104	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	23
105	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	23
106	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	24
107	4.2.4 ECP Profile Schema Usage.....	24
108	4.2.4.1 PAOS Request Header Block: SP to ECP.....	25

109	4.2.4.2 ECP Request Header Block : SP to ECP.....	25
110	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	26
111	4.2.4.4 ECP Response Header Block : IdP to ECP.....	27
112	4.2.4.5 PAOS Response Header Block : ECP to SP.....	28
113	4.2.5 Security Considerations.....	28
114	4.3 Identity Provider Discovery Profile.....	29
115	4.3.1 Common Domain Cookie.....	29
116	4.3.2 Setting the Common Domain Cookie.....	29
117	4.3.3 Obtaining the Common Domain Cookie.....	29
118	4.4 Single Logout Profile.....	30
119	4.4.1 Required Information.....	30
120	4.4.2 Profile Overview.....	30
121	4.4.3 Profile Description.....	32
122	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	32
123	4.4.3.2 Identity Provider Determines Session Participants.....	33
124	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	33
125	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	33
126	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	34
127	4.4.4 Use of Single Logout Protocol.....	34
128	4.4.4.1 <LogoutRequest> Usage.....	34
129	4.4.4.2 <LogoutResponse> Usage.....	34
130	4.4.5 Use of Metadata.....	34
131	4.5 Name Identifier Management Profile.....	35
132	4.5.1 Required Information.....	35
133	4.5.2 Profile Overview.....	35
134	4.5.3 Profile Description.....	36
135	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	36
136	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	36
137	4.5.4 Use of Name Identifier Management Protocol.....	37
138	4.5.4.1 <ManageNameIDRequest> Usage.....	37
139	4.5.4.2 <ManageNameIDResponse> Usage.....	37
140	4.5.5 Use of Metadata.....	37
141	5 Artifact Resolution Profile.....	39
142	5.1 Required Information.....	39
143	5.2 Profile Overview.....	39
144	5.3 Profile Description.....	40
145	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	40
146	5.3.2 <ArtifactResponse> issued by Responding Entity.....	40
147	5.4 Use of Artifact Resolution Protocol.....	40
148	5.4.1 <ArtifactResolve> Usage.....	40
149	5.4.2 <ArtifactResponse> Usage.....	40
150	5.5 Use of Metadata.....	41
151	6 Assertion Query/Request Profile.....	42
152	6.1 Required Information.....	42
153	6.2 Profile Overview.....	42
154	6.3 Profile Description.....	43

155	6.3.1 Query/Request issued by Requesting Entity.....	43
156	6.3.2 <Response> issued by SAML Authority.....	43
157	6.4 Use of Query/Request Protocol.....	43
158	6.4.1 Query/Request Usage.....	43
159	6.4.2 <Response> Usage.....	43
160	6.5 Use of Metadata.....	43
161	7 Name Identifier Mapping Profile.....	45
162	7.1 Required Information.....	45
163	7.2 Profile Overview.....	45
164	7.3 Profile Description.....	46
165	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	46
166	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	46
167	7.4 Use of Name Identifier Mapping Protocol.....	46
168	7.4.1 <NameIDMappingRequest> Usage.....	46
169	7.4.2 <NameIDMappingResponse> Usage.....	46
170	7.4.2.1 Limiting Use of Mapped Identifier.....	47
171	7.5 Use of Metadata.....	47
172	8 SAML Attribute Profiles.....	48
173	8.1 Basic Attribute Profile.....	48
174	8.1.1 Required Information.....	48
175	8.1.2 SAML Attribute Naming.....	48
176	8.1.2.1 Attribute Name Comparison.....	48
177	8.1.3 Profile-Specific XML Attributes.....	48
178	8.1.4 SAML Attribute Values.....	48
179	8.1.5 Example.....	48
180	8.2 X.500/LDAP Attribute Profile.....	48
181	8.2.1 Required Information.....	49
182	8.2.2 SAML Attribute Naming.....	49
183	8.2.2.1 Attribute Name Comparison.....	49
184	8.2.3 Profile-Specific XML Attributes.....	49
185	8.2.4 SAML Attribute Values.....	49
186	8.2.5 Profile-Specific Schema.....	50
187	8.2.6 Example.....	51
188	8.3 UUID Attribute Profile.....	51
189	8.3.1 Required Information.....	51
190	8.3.2 UUID and GUID Background.....	51
191	8.3.3 SAML Attribute Naming.....	51
192	8.3.3.1 Attribute Name Comparison.....	52
193	8.3.4 Profile-Specific XML Attributes.....	52
194	8.3.5 SAML Attribute Values.....	52
195	8.3.6 Example.....	52
196	8.4 DCE PAC Attribute Profile.....	52
197	8.4.1 Required Information.....	52
198	8.4.2 PAC Description.....	52

199	8.4.3 SAML Attribute Naming.....	53
200	8.4.3.1 Attribute Name Comparison.....	53
201	8.4.4 Profile-Specific XML Attributes.....	53
202	8.4.5 SAML Attribute Values.....	53
203	8.4.6 Attribute Definitions.....	54
204	8.4.6.1 Realm.....	54
205	8.4.6.2 Principal.....	54
206	8.4.6.3 Primary Group.....	54
207	8.4.6.4 Groups.....	54
208	8.4.6.5 Foreign Groups.....	55
209	8.4.7 Example.....	55
210	8.5 XACML Attribute Profile.....	56
211	8.5.1 Required Information.....	56
212	8.5.2 SAML Attribute Naming.....	56
213	8.5.2.1 Attribute Name Comparison.....	56
214	8.5.3 Profile-Specific XML Attributes.....	56
215	8.5.4 SAML Attribute Values.....	56
216	8.5.5 Profile-Specific Schema.....	56
217	8.5.6 Example.....	57
218	9 References.....	58

219

# 1 Introduction

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

223 A separate specification ([SAMLCore]) defines the SAML assertions and request-response protocol  
224 messages themselves, and another ([SAMLBind]) defines bindings of SAML protocol messages to  
225 underlying communications and messaging protocols.

## 1.1 Profile Concepts

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

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

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

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

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

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

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

## 1.2 Notation

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

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

261 `Listings of productions or other normative code appear like this.`

262 `Example code listings appear like this.`

263 **Note:** Non-normative notes and explanations appear like this.

264 Conventional XML namespace prefixes are used throughout this specification to stand for their respective  
265 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].
ldapprof :	urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP	This is the SAML V2.0 X.500/LDAP attribute profile namespace, specified in this document and in a schema [SAMLLDAP-xsd].
xacmlprof :	urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML	This is the SAML V2.0 LDAP 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.

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

269

## 2 Specification of Additional Profiles

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

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

### 2.1 Guidelines for Specifying Profiles

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

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

### 2.2 Guidelines for Specifying Attribute Profiles

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

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

310 use when processing attributes, queries, etc.

311 5. Syntax and restrictions on values acceptable in the SAML <AttributeValue> element, including  
312 whether the `xsi:type` XML attribute can or should be used.

313

## 3 Confirmation Method Identifiers

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

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

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

### 3.1 Holder of Key

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

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

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

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

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

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

### 3.2 Sender Vouches

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

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

357 **3.3 Bearer**

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

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

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

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

---

## 371 4 SSO Profiles of SAML

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

### 379 4.1 Web Browser SSO Profile

380 In the scenario supported by the web browser SSO profile, a web user either accesses a resource at a

381 service provider, or accesses an identity provider such that the service provider and desired resource are

382 understood or implicit. The web user authenticates (or has already authenticated) to the identity provider,

383 which then produces an authentication assertion (possibly with input from the service provider) and the

384 service provider consumes the assertion to establish a security context for the web user. During this

385 process, a name identifier might also be established between the providers for the principal, subject to the

386 parameters of the interaction and the consent of the parties.

387 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction

388 with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.

389 It is assumed that the user is using a standard commercial browser and can authenticate to the identity

390 provider by some means outside the scope of SAML.

#### 391 4.1.1 Required Information

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

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

394 **SAML Confirmation Method Identifiers:** The SAML V2.0 "bearer" confirmation method identifier,

395 urn:oasis:names:tc:SAML:2.0:cm:bearer, is used by this profile.

396 **Description:** Given below.

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

#### 398 4.1.2 Profile Overview

399 Figure 1 illustrates the basic template for achieving SSO. The following steps are described by the profile.

400 Within an individual step, there may be one or more actual message exchanges depending on the binding

401 used for that step and other implementation-dependent behavior.

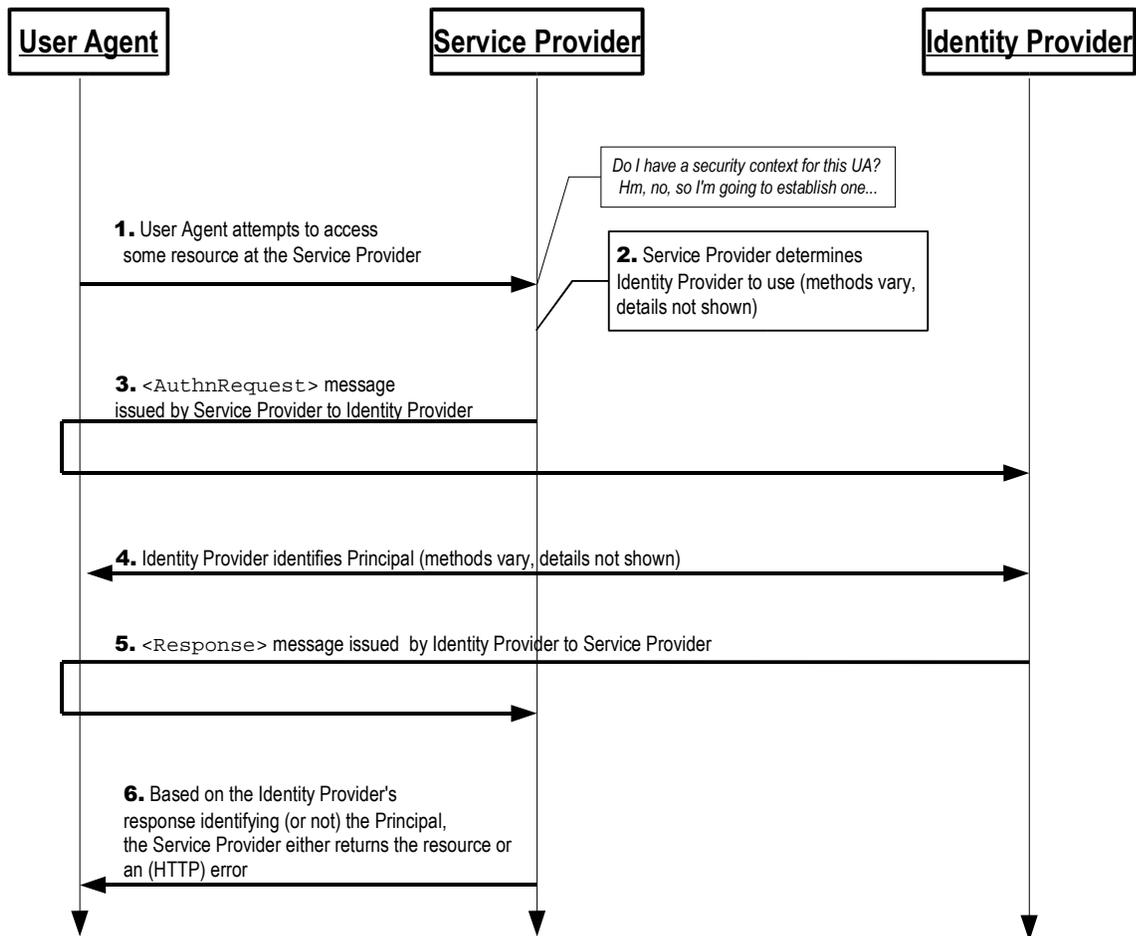


Figure 1

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

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

405 **2. Service Provider Determines Identity Provider**

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

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

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

414 **4. Identity Provider identifies Principal**

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

418 **5. Identity Provider issues <Response> to Service Provider**

419 In step 5, the identity provider issues a <Response> message to be delivered by the user agent  
420 to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer

421 the message to the service provider through the user agent. The message may indicate an error,  
422 or will include (at least) an authentication assertion. The HTTP Redirect binding MUST NOT be  
423 used, as the response will typically exceed the URL length permitted by most user agents.

## 424 **6. Service Provider grants or denies access to Principal**

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

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

## 430 **4.1.3 Profile Description**

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

### 433 **Single Sign-On Service**

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

### 436 **Assertion Consumer Service**

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

## 439 **4.1.3.1 HTTP Request to Service Provider**

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

## 446 **4.1.3.2 Service Provider Determines Identity Provider**

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

## 453 **4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider**

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

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

466 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or TLS  
467 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <AuthnRequest> message MAY  
468 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also  
469 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.  
470 The identity provider MUST process the <AuthnRequest> message as described in [SAMLCore]. This  
471 may constrain the subsequent interactions with the user agent, for example if the `IsPassive` attribute is  
472 included.

#### 473 **4.1.3.4 Identity Provider Identifies Principal**

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

#### 481 **4.1.3.5 Identity Provider Issues <Response> to Service Provider**

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

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

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

495 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 ([SSL3]) or TLS  
496 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <Assertion> element(s) in the  
497 <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-  
498 Artifact binding is used.

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

#### 501 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

502 To complete the profile, the service provider processes the <Response> and <Assertion>(s) and  
503 grants or denies access to the resource. The service provider MAY establish a security context with the  
504 user agent using any session mechanism it chooses. Any subsequent use of the <Assertion>(s)  
505 provided are at the discretion of the service provider and other relying parties, subject to any restrictions  
506 on use contained within them.

#### 507 **4.1.4 Use of Authentication Request Protocol**

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

#### 512 4.1.4.1 <AuthnRequest> Usage

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

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

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

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

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

#### 531 4.1.4.2 <Response> Usage

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

- 535 • The <Issuer> element MAY be omitted, but if present it MUST contain the unique identifier of the  
536 issuing identity provider; the Format attribute MUST be omitted or have a value of  
537 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 538 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the  
539 unique identifier of the issuing identity provider; the Format attribute MUST be omitted or have a value  
540 of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 541 • The set of one or more assertions MUST contain at least one <AuthnStatement> that reflects the  
542 authentication of the principal to the identity provider.
- 543 • At least one assertion containing an <AuthnStatement> MUST contain a <Subject> element with  
544 at least one <SubjectConfirmation> element containing a Method of  
545 urn:oasis:names:tc:SAML:2.0:cm:bearer. If the identity provider supports the Single Logout profile,  
546 defined in Section 4.4, any such authentication statements MUST include a SessionIndex attribute  
547 to enable per-session logout requests by the service provider.
- 548 • Any bearer <SubjectConfirmationData> elements MUST contain a Recipient attribute  
549 containing the service provider's assertion consumer service URL and a NotOnOrAfter attribute that  
550 limits the window during which the assertion can be delivered. It MAY contain an Address attribute  
551 limiting the client address from which the assertion can be delivered. It MUST NOT contain a  
552 NotBefore attribute. If the containing message is in response to an <AuthnRequest>, then the  
553 InResponseTo attribute MUST match the request's ID.
- 554 • Other statements and confirmation methods MAY be included in the assertion(s) at the discretion of  
555 the identity provider. In particular, <AttributeStatement> elements MAY be included. The  
556 <AuthnRequest> MAY contain an AttributeConsumingServiceIndex XML attribute  
557 referencing information about desired or required attributes in [SAMLMeta]. The identity provider MAY  
558 ignore this, or send other attributes at its discretion.
- 559 • The assertion(s) containing a bearer subject confirmation MUST contain an  
560 <AudienceRestriction> including the service provider's unique identifier as an <Audience>.

- 561 • Other conditions (and other <Audience> elements) MAY be included as requested by the service  
562 provider or at the discretion of the identity provider. (Of course, all such conditions MUST be  
563 understood by and accepted by the service provider in order for the assertion to be considered valid.)  
564 The identity provider is NOT obligated to honor the requested set of <Conditions> in the  
565 <AuthnRequest>, if any.

#### 566 **4.1.4.3 <Response> Message Processing Rules**

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

- 568 • Verify any signatures present on the assertion(s) or the response
- 569 • Verify that the `Recipient` attribute in any bearer <SubjectConfirmationData> matches the  
570 assertion consumer service URL to which the <Response> or artifact was delivered
- 571 • Verify that the `NotOnOrAfter` attribute in any bearer <SubjectConfirmationData> has not  
572 passed, subject to allowable clock skew between the providers
- 573 • Verify that the `InResponseTo` attribute in the bearer <SubjectConfirmationData> equals the ID  
574 of its original <AuthnRequest> message, unless the response is unsolicited (see Section 4.5) in  
575 which case the attribute MUST NOT be present
- 576 • Verify that any assertions relied upon are valid in other respects

577 If any bearer <SubjectConfirmationData> includes an `Address` attribute, the service provider MAY  
578 check the user agent's client address against it.

579 Any assertion which is not valid, or whose subject confirmation requirements cannot be met SHOULD be  
580 discarded and SHOULD NOT be used to establish a security context for the principal.

581 If an <AuthnStatement> used to establish a security context for the principal contains a  
582 `SessionNotOnOrAfter` attribute, the security context SHOULD be discarded once this time is reached,  
583 unless the service provider reestablishes the principal's identity by repeating the use of this profile.

#### 584 **4.1.4.4 Artifact-Specific <Response> Message Processing Rules**

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

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

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

#### 591 **4.1.4.5 POST-Specific Processing Rules**

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

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

#### 597 **4.1.5 Unsolicited Responses**

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

600 An unsolicited <Response> MUST NOT contain an `InResponseTo` attribute, nor should any bearer  
601 <SubjectConfirmationData> elements contain one. If metadata as specified in [SAMLMeta] is used,  
602 the <Response> or artifact SHOULD be delivered to the <md:AssertionConsumerService> endpoint

603 of the service provider designated as the default.  
604 Of special mention is that the identity provider SHOULD include a binding-specific "RelayState" parameter  
605 that indicates, based on mutual agreement with the service provider, how to handle subsequent  
606 interactions with the user agent. This MAY be the URL of a resource at the service provider.

#### 607 4.1.6 Use of Metadata

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

611 The `<md:IDPDescriptor>` element's `WantAuthnRequestsSigned` attribute MAY be used by an  
612 identity provider to document a requirement that requests be signed. The `<md:SPDescriptor>`  
613 element's `AuthnRequestsSigned` attribute MAY be used by a service provider to document the  
614 intention to sign all of its requests.

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

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

624 The `<md:SPDescriptor>` element's `WantAssertionsSigned` attribute MAY be used by a service  
625 provider to document a requirement that assertions delivered with this profile be signed. This is in addition  
626 to any requirements for signing imposed by the use of a particular binding.

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

629 The `<md:AttributeConsumerDescriptor>` element MAY be used to document the service provider's  
630 need or desire for SAML attributes to be delivered along with authentication information. The actual  
631 inclusion of attributes is of course at the discretion of the identity provider. One or more  
632 `<md:AttributeConsumingService>` elements MAY be included in its metadata, each with an `index`  
633 attribute to distinguish different services that MAY be specified by reference in the `<AuthnRequest>`  
634 message. The `isDefault` attribute is used to specify a default set of attribute requirements.

#### 635 4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

648 **Note:** The means by which a p[ri]ncipal authenticates with an identity provider is outside of the  
649 scope of SAML.

## 650 4.2.1 Required Information

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

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

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

656 **Description:** Given below.

657 **Updates:** None.

## 658 4.2.2 Profile Overview

659 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and  
660 service providers and identity providers. It is a specific application of the SSO profile described in Section  
661 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the  
662 rules specified in Section 4.1 MUST be observed.

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

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

666 This allows a service provider to make an authentication request to the ECP without the need to know  
667 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section  
668 4.1).

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

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

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

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

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

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

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

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

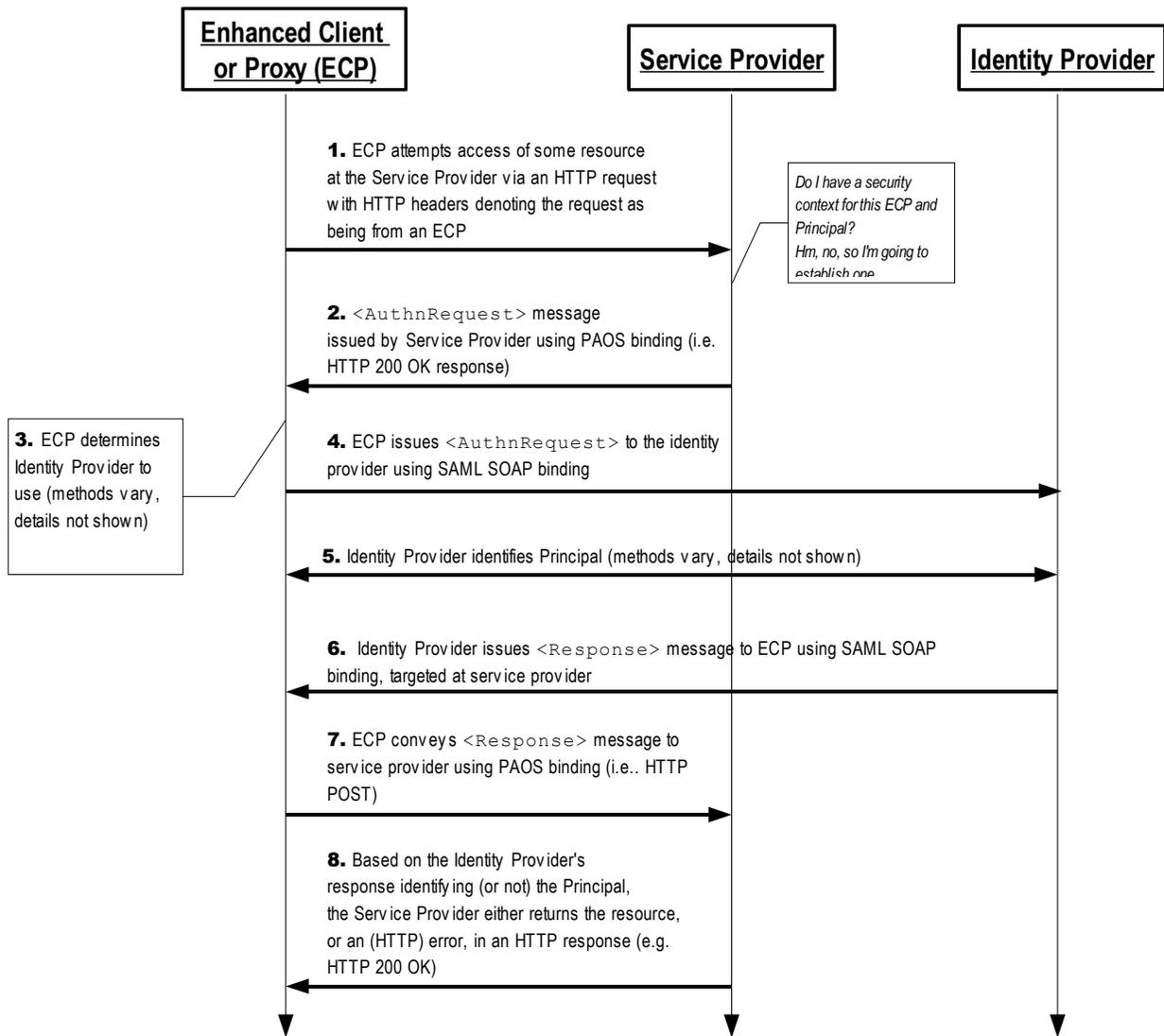


Figure 2

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

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

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

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

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

706 **3. ECP Determines Identity Provider**

707 In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication

708 request protocol that supports its preferred binding. The means by which this is accomplished is  
709 implementation-dependent. The ECP MAY use the SAML identity provider discovery profile  
710 described in Section 4.3.

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

712 In step 4, the ECP conveys the <AuthnRequest> to the identity provider identified in step 3  
713 using the SAML SOAP binding [SAMLBind].

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

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

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

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

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

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

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

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

### 729 **4.2.3 Profile Description**

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

#### 731 **4.2.3.1 ECP issues HTTP Request to Service Provider**

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

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

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

```
742 GET /index HTTP/1.1  
743 Host: identity-service.example.com  
744 Accept: text/html; application/vnd.paos+xml  
745 PAOS: ver='urn:liberty:paos:2003-08' ;  
746 'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
```

#### 747 **4.2.3.2 Service Provider Issues <AuthnRequest> to ECP**

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

752 The SOAP envelope MUST contain:

- 753 1. An `<AuthnRequest>` element in the SOAP body, intended for the ultimate SOAP recipient, the  
754 identity provider.
- 755 2. A PAOS SOAP header block targeted at the ECP using the SOAP `actor` value of  
756 `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control  
757 information such as the URL to which to send the response in this solicit-response message  
758 exchange pattern.
- 759 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor  
760 `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines  
761 information related to the authentication request that the ECP may need to process it, such as a list  
762 of identity providers acceptable to the service provider, whether the ECP may interact with the  
763 principal through the client, and the service provider's human-readable name that may be displayed  
764 to the principal.

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

#### 768 **4.2.3.3 ECP Determines Identity Provider**

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

#### 770 **4.2.3.4 ECP issues `<AuthnRequest>` to Identity Provider**

771 The ECP MUST remove the PAOS, ECP RelayState, and ECP Request header blocks before passing the  
772 `<AuthnRequest>` message on to the identity provider, using the SAML SOAP binding.

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

775 Prior to or subsequent to this step, the identity provider MUST establish the identity of the principal by  
776 some means, or it MUST return an error `<Response>` in step 4, described below.

#### 777 **4.2.3.5 Identity Provider Identifies Principal**

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

#### 785 **4.2.3.6 Identity Provider issues `<Response>` to ECP, targeted at service provider**

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

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

#### 793 **4.2.3.7 ECP Conveys `<Response>` Message to Service Provider**

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

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

802 The RelayState header block value is typically provided by the service provider to the ECP with its request,  
803 but if the identity provider is producing an unsolicited response (without having received a corresponding  
804 SAML request), then it SHOULD include a RelayState header block that indicates, based on mutual  
805 agreement with the service provider, how to handle subsequent interactions with the ECP. This MAY be  
806 the URL of a resource at the service provider.

807 If the service provider included a RelayState SOAP header block in its request to the ECP, or if the identity  
808 provider included a RelayState SOAP header block with its response, then the ECP MUST include an  
809 identical header block with the SAML response sent to the service provider. The service provider's value  
810 for this header block (if any) MUST take precedence.

#### 811 4.2.3.8 Service Provider Grants or Denies Access to Principal

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

#### 817 4.2.4 ECP Profile Schema Usage

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

```
820 <schema
821   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
822   xmlns="http://www.w3.org/2001/XMLSchema"
823   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
824   xmlns:samplp="urn:oasis:names:tc:SAML:2.0:protocol"
825   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
826   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
827   elementFormDefault="unqualified"
828   attributeFormDefault="unqualified"
829   blockDefault="substitution"
830   version="2.0">
831   <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
832     schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
833   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
834     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
835   <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
836     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
837
838   <element name="Request" type="ecp:RequestType"/>
839   <complexType name="RequestType">
840     <sequence>
841       <element ref="saml:Issuer"/>
842       <element ref="samplp:IDPList" minOccurs="0"/>
843     </sequence>
844     <attribute ref="SOAP-ENV:mustUnderstand" use="required"/>
845     <attribute ref="SOAP-ENV:actor" use="required"/>
846     <attribute name="ProviderName" type="string" use="optional"/>
847     <attribute name="IsPassive" type="boolean" use="optional"/>
848   </complexType>
849
850   <element name="Response" type="ecp:ResponseType"/>
851   <complexType name="ResponseType">
```

```

850         <attribute ref="SOAP-ENV:mustUnderstand" use="required"/>
851         <attribute ref="SOAP-ENV:actor" use="required"/>
852         <attribute name="AssertionConsumerServiceURL" type="anyURI"
853 use="required"/>
854     </complexType>

855     <element name="RelayState" type="ecp:RelayStateType"/>
856     <complexType name="RelayStateType">
857         <simpleContent>
858             <extension base="string">
859                 <attribute ref="SOAP-ENV:mustUnderstand"
860 use="required"/>
861                 <attribute ref="SOAP-ENV:actor" use="required"/>
862             </extension>
863         </simpleContent>
864     </complexType>
865 </schema>

```

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

#### 867 **4.2.4.1 PAOS Request Header Block: SP to ECP**

868 The PAOS Request header block signals the use of PAOS processing and includes the following  
869 attributes:

870 `responseConsumerURL` [Required]

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

876 `service` [Required]

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

879 `SOAP-ENV:mustUnderstand` [Required]

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

882 `SOAP-ENV:actor` [Required]

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

884 `messageID` [Optional]

885 Allows optional response correlation. It MAY be used in this profile, but is NOT required, since this  
886 functionality is provided by the SAML protocol layer, via the `ID` attribute in the `<AuthnRequest>` and  
887 the `InResponseTo` attribute in the `<Response>`.

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

#### 889 **4.2.4.2 ECP Request Header Block : SP to ECP**

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

893 `SOAP-ENV:mustUnderstand` [Required]

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

896 SOAP-ENV:actor [Required]

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

898 ProviderName [Optional]

899 A human-readable name for the requesting service provider.

900 IsPassive [Optional]

901 A boolean value. If `true`, the identity provider and the client itself MUST NOT take control of the user  
902 interface from the request issuer and interact with the principal in a noticeable fashion. If a value is not  
903 provided, the default is `true`.

904 <saml:Issuer> [Required]

905 This element MUST contain the unique identifier of the requesting service provider; the `Format`  
906 attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

907 <samlp:IDPList> [Optional]

908 Optional list of identity providers that the service provider recognizes and from which the ECP may  
909 choose to service the request. See [SAMLCore] for details on the content of this element.

#### 910 4.2.4.3 ECP RelayState Header Block: SP to ECP

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

914 SOAP-ENV:mustUnderstand [Required]

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

916 SOAP-ENV:actor [Required]

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

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

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

```
924 <SOAP-ENV:Envelope
925     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
926     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
927     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
928   <SOAP-ENV:Header>
929     <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
930       responseConsumerURL="http://identity-service.example.com/abc"
931       messageID="6c3a4f8b9c2d" SOAP-
932 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
933 ENV:mustUnderstand="1"
934       service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
935     </paos:Request>
936     <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
937       SOAP-ENV:mustUnderstand="1" SOAP-
938 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
939       ProviderName="Service Provider X" IsPassive="0">
940     <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
941     <samlp:IDPList>
942     <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
943       Name="Identity Provider X"
```

```

944         Loc="https://IdentityProvider.example.com/saml2/sso"
945     </samlp:IDPEntry>
946     <samlp:GetComplete>
947         https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
948     </samlp:GetComplete>
949 </samlp:IDPList>
950 </ecp:Request>
951 <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
952     SOAP-ENV:mustUnderstand="1" SOAP-
953 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
954     ...
955 </ecp:RelayState>
956 </SOAP-ENV:Header>
957 <SOAP-ENV:Body>
958     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
959 </SOAP-ENV:Body>
960 </SOAP-ENV:Envelope>

```

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

```

964 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
965     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
966     <SOAP-ENV:Body>
967         <samlp:AuthnRequest> ... </samlp:AuthnRequest>
968     </SOAP-ENV:Body>
969 </SOAP-ENV:Envelope>

```

#### 970 4.2.4.4 ECP Response Header Block : IdP to ECP

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

973 SOAP-ENV:mustUnderstand [Required]

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

976 SOAP-ENV:actor [Required]

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

978 AssertionConsumerServiceURL [Required]

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

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

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

988 The ECP Response SOAP header has no element content.

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

```

990 <SOAP-ENV:Envelope
991     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
992     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
993     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

```

```

994     <SOAP-ENV:Header>
995         <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
996 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
997 AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_
998 consumer"/>
999     </SOAP-ENV:Header>
1000    <SOAP-ENV:Body>
1001        <samlp:Response> ... </samlp:Response>
1002    </SOAP-ENV:Body>
1003 </SOAP-ENV:Envelope>

```

#### 1004 4.2.4.5 PAOS Response Header Block : ECP to SP

1005 The PAOS Response header block includes the following attributes:

1006 SOAP-ENV:mustUnderstand [Required]

1007 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not  
1008 understood.

1009 SOAP-ENV:actor [Required]

1010 The value MUST be http://schemas.xmlsoap.org/soap/actor/next.

1011 refToMessageID [Optional]

1012 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)  
1013 MUST be added by the ECP if the corresponding PAOS request specified the messageID attribute.  
1014 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>  
1015 correlation.

1016 The PAOS Response SOAP header has no element content.

1017 Following is an example of an ECP-to-SP response.

```

1018 <SOAP-ENV:Envelope
1019     xmlns:paos="urn:liberty:paos:2003-08"
1020     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1021     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1022     <SOAP-ENV:Header>
1023         <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1024 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1025 ENV:mustUnderstand="1"/>
1026         <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1027 SOAP-ENV:mustUnderstand="1" SOAP-
1028 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1029             ...
1030         </ecp:RelayState>
1031     </SOAP-ENV:Header>
1032     <SOAP-ENV:Body>
1033         <samlp:Response> ... </samlp:Response>
1034     </SOAP-ENV:Body>
1035 </SOAP-ENV:Envelope>

```

#### 1036 4.2.5 Security Considerations

1037 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,  
1038 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP  
1039 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security  
1040 countermeasures appropriate to that binding are used.

1041 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the  
1042 use of SSL/TLS over every HTTP exchange with the client.

1043 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS  
1044 authentication.

1045 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated  
1046 session.

## 1047 **4.3 Identity Provider Discovery Profile**

1048 This section defines a profile by which a service provider can discover which identity providers a principal  
1049 is using with the Web Browser SSO profile. In deployments having more than one identity provider,  
1050 service providers need a means to discover which identity provider(s) a principal uses. The discovery  
1051 profile relies on a cookie that is written in a domain that is common between identity providers and service  
1052 providers in a deployment. The domain that the deployment predetermines is known as the common  
1053 domain in this profile, and the cookie containing the list of identity providers is known as the common  
1054 domain cookie.

1055 Which entities host web servers in the common domain is a deployment issue and is outside the scope of  
1056 this profile.

### 1057 **4.3.1 Common Domain Cookie**

1058 The name of the cookie MUST be `_saml_idp`. The format of the cookie value MUST be a set of one or  
1059 more base-64 encoded URI values separated by a single space character. Each URI is the unique  
1060 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCORE]. The final set of values is then  
1061 URL encoded.

1062 The common domain cookie writing service (see below) SHOULD append the identity provider's unique  
1063 identifier to the list. If the identifier is already present in the list, it MAY remove and append it when  
1064 authentication of the principal occurs. The intent is that the most recently established identity provider  
1065 session is the last one in the list.

1066 The cookie MUST be set with no Path prefix or a Path prefix of `"/`. The Domain MUST be set to  
1067 `"[common-domain]"` where `[common-domain]` is the common domain established within the deployment  
1068 for use with this profile. The cookie MUST be marked as secure.

1069 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCOOKIE]. The cookie MAY  
1070 be either session-only or persistent. This choice may be made within a deployment, but should apply  
1071 uniformly to all identity providers in the deployment.

### 1072 **4.3.2 Setting the Common Domain Cookie**

1073 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by  
1074 which the identity provider sets the cookie are implementation-specific so long as the cookie is  
1075 successfully set with the parameters given above. One possible implementation strategy follows and  
1076 should be considered non-normative. The identity provider may:

- 1077 • Have previously established a DNS and IP alias for itself in the common domain.
- 1078 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL  
1079 scheme. The structure of the URL is private to the implementation and may include session  
1080 information needed to identify the user-agent.
- 1081 • Set the cookie on the redirected user agent using the parameters specified above.
- 1082 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

### 1083 **4.3.3 Obtaining the Common Domain Cookie**

1084 When a service provider needs to discover which identity providers a principal uses, it invokes an  
1085 exchange designed to present the common domain cookie to the service provider after it is read by an  
1086 HTTP server in the common domain.

1087 If the HTTP server in the common domain is operated by the service provider or if other arrangements are  
1088 in place, the service provider MAY utilize the HTTP server in the common domain to relay its  
1089 `<AuthnRequest>` to the identity provider for an optimized single sign-on process.

1090 The specific means by which the service provider reads the cookie are implementation-specific so long as  
1091 it is able to cause the user agent to present cookies that have been set with the parameters given in  
1092 Section 4.3.1. One possible implementation strategy is described as follows and should be considered  
1093 non-normative. Additionally, it may be sub-optimal for some applications.  
1094 • Have previously established a DNS and IP alias for itself in the common domain.  
1095 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL  
1096 scheme. The structure of the URL is private to the implementation and may include session  
1097 information needed to identify the user-agent.  
1098 • Set the cookie on the redirected user agent using the parameters specified above.  
1099 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

## 1100 4.4 Single Logout Profile

1101 Once a principal has authenticated to an identity provider, the authenticating entity may establish a  
1102 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-  
1103 specific means). The identity provider may subsequently issue assertions to service providers or other  
1104 relying parties, based on this authentication event; a relying party may use this to establish *its own* session  
1105 with the user.

1106 In such a situation, the identity provider can act as a session authority and the relying parties as session  
1107 participants. At some later time, the principal may wish to terminate his or her session either with an  
1108 individual session participant, or with all session participants in a given session managed by the session  
1109 authority. The former case is considered out of scope of this specification. The latter case, however, may  
1110 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1111 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this  
1112 "global" session either by contacting the session authority, or an individual session participant. Also note  
1113 that an identity provider acting as a session authority may *itself* act as a session participant in situations in  
1114 which it is the relying party for another identity provider's assertions regarding that principal.

1115 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or  
1116 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A  
1117 front-channel binding may be required, for example, in cases in which a principal's session state exists  
1118 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the  
1119 session participant or session authority is required.

### 1120 4.4.1 Required Information

1121 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1122 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1123 **Description:** Given below.

1124 **Updates:** None

### 1125 4.4.2 Profile Overview

1126 Figure 3 illustrates the basic template for achieving single logout:

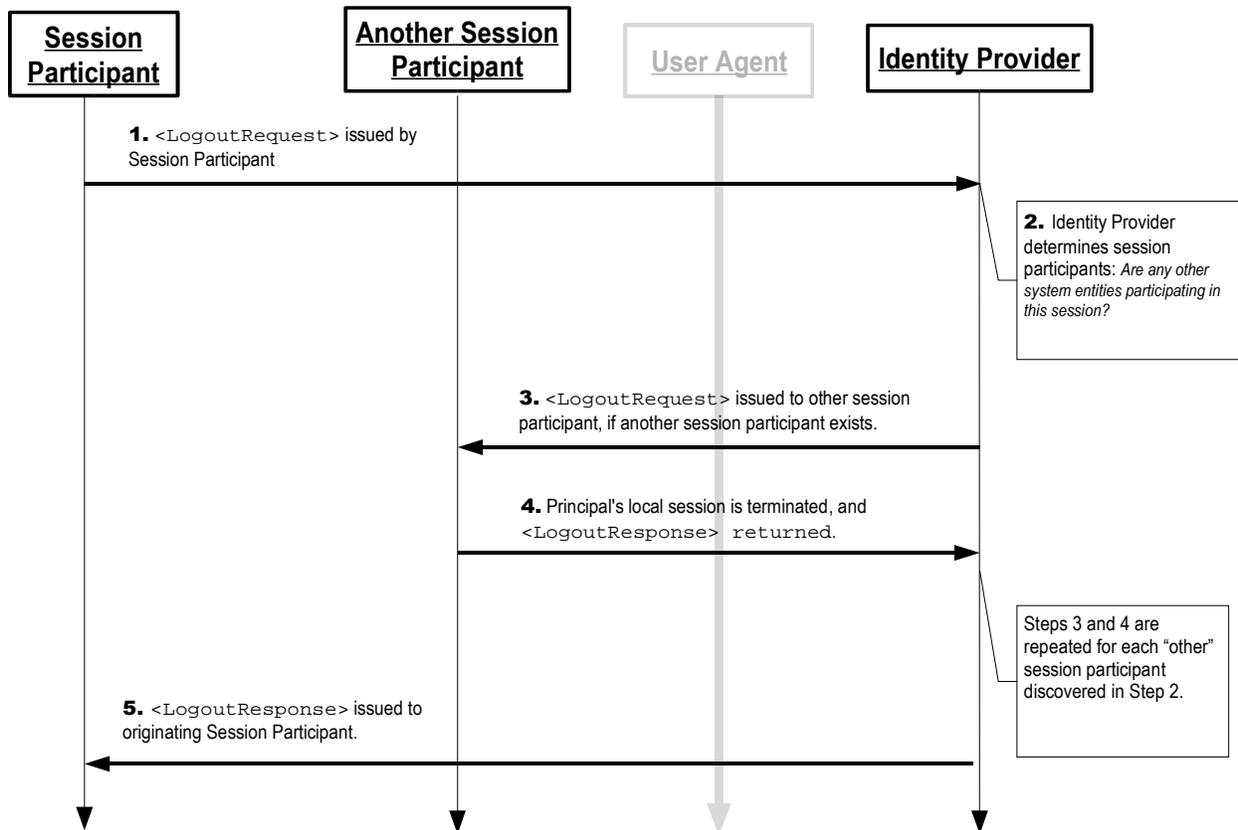


Figure 3

1127 The grayed-out user agent illustrates that the message exchange may pass through the user agent or  
 1128 may be a direct exchange between system entities, depending on the SAML binding used to implement  
 1129 the profile.

1130 The following steps are described by the profile. Within an individual step, there may be one or more  
 1131 actual message exchanges depending on the binding used for that step and other implementation-  
 1132 dependent behavior.

1133 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1134 In step 1, the session participant initiates single logout and terminates a principal's session(s) by  
 1135 sending a <LogoutRequest> message to the identity provider from whom it received the  
 1136 corresponding authentication assertion. The request may be sent directly to the identity provider  
 1137 or sent indirectly through the user agent.

1138 **2. Identity Provider determines Session Participants**

1139 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if  
 1140 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If  
 1141 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4  
 1142 are repeated for each session participant identified.

1143 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1144 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or  
 1145 session authority related to one or more of the session(s) being terminated. The request may be  
 1146 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the  
 1147 request in step 1).

1148 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1149 In step 4, a session participant or session authority terminates the principal's session(s) as

1150 directed by the request (if possible) and returns a `<LogoutResponse>` to the identity provider.  
1151 The response may be returned directly to the identity provider or indirectly through the user agent  
1152 (if consistent with the form of the request in step 3).

#### 1153 **5. Identity Provider issues `<LogoutResponse>` to Session Participant**

1154 In step 5, the identity provider issues a `<LogoutResponse>` message to the original requesting  
1155 session participant. The response may be returned directly to the session participant or indirectly  
1156 through the user agent (if consistent with the form of the request in step 1).

1157 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a  
1158 `<LogoutRequest>` to all session participants, also skipping step 5.

### 1159 **4.4.3 Profile Description**

1160 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity  
1161 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

#### 1162 **Single Logout Service**

1163 This is the single logout protocol endpoint at an identity provider or session participant to which the  
1164 `<LogoutRequest>` or `<LogoutResponse>` messages (or an artifact representing them) are  
1165 delivered. The same or different endpoints MAY be used for requests and responses.

#### 1166 **4.4.3.1 `<LogoutRequest>` Issued by Session Participant to Identity Provider**

1167 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it  
1168 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it  
1169 received from the identity provider. If multiple identity providers are involved, then the profile MUST be  
1170 repeated independently for each one.

1171 To initiate the profile, the session participant issues a `<LogoutRequest>` message to the identity  
1172 provider's single logout service request endpoint containing one or more applicable `<SessionIndex>`  
1173 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to  
1174 determine the location of this endpoint and the bindings supported by the identity provider.

#### 1175 **Synchronous Bindings (Back-Channel)**

1176 The session participant MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to  
1177 send the request directly to the identity provider. The identity provider would then propagate any  
1178 required logout messages to additional session participants as required using a synchronous binding.  
1179 The requester MUST authenticate itself to the identity provider, either by signing the  
1180 `<LogoutRequest>` or using any other binding-supported mechanism.

#### 1181 **Asynchronous Bindings (Front-Channel)**

1182 Alternatively, the session participant MAY (if the principal's user agent is present) use an  
1183 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the  
1184 request to the identity provider through the user agent.

1185 If the HTTP Redirect or POST binding is used, then the `<LogoutRequest>` message is delivered to  
1186 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile  
1187 defined in Section 5 is used by the identity provider, which makes a callback to the session participant  
1188 to retrieve the `<LogoutRequest>` message, using for example the SOAP binding.

1189 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or  
1190 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The `<LogoutRequest>`  
1191 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,  
1192 if used, also provides for an alternate means of authenticating the request issuer when the artifact is  
1193 dereferenced.

1194 Each of these bindings provide a RelayState mechanism that the session participant MAY use to  
1195 associate the profile exchange with the original request. The session participant SHOULD reveal as  
1196 little information as possible in the RelayState value unless the use of the profile does not require such

1197 privacy measures.

1198 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

#### 1199 **4.4.3.2 Identity Provider Determines Session Participants**

1200 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>  
1201 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the  
1202 principal identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1203 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being  
1204 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2  
1205 of [SAMLCore].

#### 1206 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1207 Participant/Authority**

1208 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or  
1209 participant in a session being terminated. The request is sent in the same fashion as described in step 1  
1210 using a SAML binding consistent with the capability of the responder and the availability of the user agent  
1211 at the identity provider.

1212 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

#### 1213 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity 1214 Provider**

1215 The session participant/authority MUST process the <LogoutRequest> message as defined in  
1216 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a  
1217 <LogoutResponse> message containing an appropriate status code to the requesting identity provider  
1218 to complete the SAML protocol exchange.

##### 1219 **Synchronous Bindings (Back-Channel)**

1220 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the  
1221 response is returned directly to complete the synchronous communication. The responder MUST  
1222 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or  
1223 using any other binding-supported mechanism.

##### 1224 **Asynchronous Bindings (Front-Channel)**

1225 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact  
1226 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to  
1227 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be  
1228 used to determine the location of this endpoint and the bindings supported by the identity provider.

1229 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is delivered to  
1230 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile  
1231 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity  
1232 to retrieve the <LogoutResponse> message, using for example the SOAP binding.

1233 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or  
1234 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutResponse>  
1235 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,  
1236 if used, also provides for an alternate means of authenticating the response issuer when the artifact is  
1237 dereferenced.

1238 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section  
1239 4.4.4.1.

#### 1240 4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant

1241 After processing the original session participant's <LogoutRequest> in step 1, or upon encountering an  
1242 error, the identity provider MUST respond to the original request with a <LogoutResponse> containing  
1243 an appropriate status code to complete the SAML protocol exchange.

1244 The response is sent to the original session participant in the same fashion as described in step 4, using a  
1245 SAML binding consistent with the binding used in the request, the capability of the responder, and the  
1246 availability of the user agent at the identity provider.

1247 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section  
1248 4.4.4.2.

#### 1249 4.4.4 Use of Single Logout Protocol

##### 1250 4.4.4.1 <LogoutRequest> Usage

1251 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;  
1252 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-  
1253 format:entity.

1254 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing  
1255 the message or using a binding-specific mechanism.

1256 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in  
1257 the authentication assertion the requester issued or received regarding the session being terminated, per  
1258 the matching rules defined in Section 3.3.4 of [SAMLCore].

1259 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the  
1260 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such  
1261 elements to indicate the termination of all of the principal's applicable sessions.

##### 1262 4.4.4.2 <LogoutResponse> Usage

1263 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1264 entity; the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-  
1265 format:entity.

1266 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1267 the message or using a binding-specific mechanism.

#### 1268 4.4.5 Use of Metadata

1269 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported  
1270 bindings and location(s) to which an entity may send requests and responses using this profile.

1271 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>  
1272 element with a use attribute of encryption to determine an appropriate encryption algorithm and  
1273 settings to use, along with a public key to use in delivering a bulk encryption key.

## 1274 4.5 Name Identifier Management Profile

1275 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged  
1276 some form of persistent identifier for a principal with a service provider, allowing them to share a common  
1277 identifier for some length of time. Subsequently, the identity provider may wish to notify the service  
1278 provider of a change in the format and/or value that it will use to identify the same principal in the future.  
1279 Alternatively the service provider may wish to attach its own "alias" for the principal in order to insure that  
1280 the identity provider will include it when communicating with it in the future about the principal. Finally, one  
1281 of the providers may wish to inform the other that it will no longer issue or accept messages using a  
1282 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management  
1283 protocol is used.

1284 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or  
1285 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A  
1286 front-channel binding may be required, for example, in cases in which direct interaction between the user  
1287 agent and the responding provider is required in order to effect the change.

### 1288 4.5.1 Required Information

1289 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1290 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1291 **Description:** Given below.

1292 **Updates:** None.

### 1293 4.5.2 Profile Overview

1294 Figure 4 illustrates the basic template for the name identifier management profile.

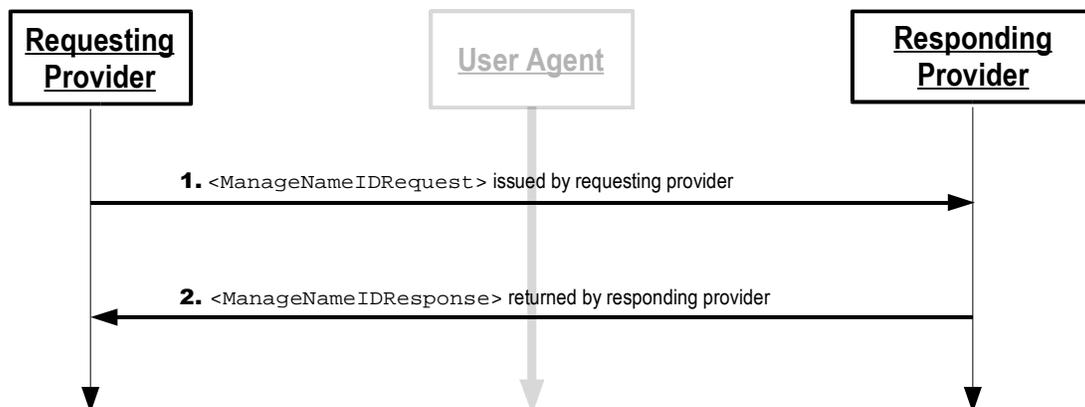


Figure 4

1295 The grayed-out user agent illustrates that the message exchange may pass through the user agent or  
1296 may be a direct exchange between system entities, depending on the SAML binding used to implement  
1297 the profile.

1298 The following steps are described by the profile. Within an individual step, there may be one or more  
1299 actual message exchanges depending on the binding used for that step and other implementation-  
1300 dependent behavior.

#### 1301 1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider

1302 In step 1, an identity or service provider initiates the profile by sending a  
1303 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.

1304 The request may be sent directly to the responding provider or sent indirectly through the user

1305 agent.

## 1306 2. <ManageNameIDResponse> issued by Responding Identity/Service Provider

1307 In step 2, the responding provider (after processing the request) issues a  
1308 <ManageNameIDResponse> message to the original requesting provider. The response may be  
1309 returned directly to the requesting provider or indirectly through the user agent (if consistent with  
1310 the form of the request in step 1).

### 1311 4.5.3 Profile Description

1312 In the descriptions below, the following is referred to:

#### 1313 Name Identifier Management Service

1314 This is the name identifier management protocol endpoint at an identity or service provider to which  
1315 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact  
1316 representing them) are delivered. The same or different endpoints MAY be used for requests and  
1317 responses.

#### 1318 4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider

1319 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another  
1320 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be  
1321 used to determine the location of this endpoint and the bindings supported by the responding provider.

##### 1322 Synchronous Bindings (Back-Channel)

1323 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to  
1324 send the request directly to the other provider. The requester MUST authenticate itself to the other  
1325 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported  
1326 mechanism.

##### 1327 Asynchronous Bindings (Front-Channel)

1328 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an  
1329 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the  
1330 request to the other provider through the user agent.

1331 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is  
1332 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution  
1333 profile defined in Section 55 is used by the other provider, which makes a callback to the requesting  
1334 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1335 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or  
1336 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The  
1337 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is  
1338 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the  
1339 request issuer when the artifact is dereferenced.

1340 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to  
1341 associate the profile exchange with the original request. The requesting provider SHOULD reveal as  
1342 little information as possible in the RelayState value unless the use of the profile does not require such  
1343 privacy measures.

1344 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section  
1345 4.5.4.1.

#### 1346 4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1347 Provider

1348 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After

1349 processing the message or upon encountering an error, the recipient MUST issue a  
1350 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider  
1351 to complete the SAML protocol exchange.

#### 1352 **Synchronous Bindings (Back-Channel)**

1353 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the  
1354 response is returned directly to complete the synchronous communication. The responder MUST  
1355 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or  
1356 using any other binding-supported mechanism.

#### 1357 **Asynchronous Bindings (Front-Channel)**

1358 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or  
1359 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through  
1360 the user agent to the requesting provider's name identifier management service response endpoint.  
1361 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings  
1362 supported by the requesting provider.

1363 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is  
1364 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact  
1365 Resolution profile defined in Section 55 is used by the requesting provider, which makes a callback to  
1366 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the  
1367 SOAP binding.

1368 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or  
1369 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The  
1370 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is  
1371 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the  
1372 response issuer when the artifact is dereferenced.

1373 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in  
1374 Section 4.5.4.2.

### 1375 **4.5.4 Use of Name Identifier Management Protocol**

#### 1376 **4.5.4.1 <ManageNameIDRequest> Usage**

1377 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;  
1378 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1379 `format:entity`.

1380 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing  
1381 the message or using a binding-specific mechanism.

#### 1382 **4.5.4.2 <ManageNameIDResponse> Usage**

1383 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1384 entity; the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1385 `format:entity`.

1386 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1387 the message or using a binding-specific mechanism.

### 1388 **4.5.5 Use of Metadata**

1389 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported  
1390 bindings and location(s) to which an entity may send requests and responses using this profile.

1391 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>  
1392 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and

1393 settings to use, along with a public key to use in delivering a bulk encryption key.

1394

## 5 Artifact Resolution Profile

1395 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding  
1396 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML  
1397 protocol messages by reference. This profile describes the use of this protocol with a synchronous  
1398 binding, such as the SOAP binding defined in [SAMLBind].

### 5.1 Required Information

1400 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1401 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1402 **Description:** Given below.

1403 **Updates:** None

### 5.2 Profile Overview

1405 The message exchange and basic processing rules that govern this profile are largely defined by Section  
1406 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to  
1407 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to  
1408 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1409 Figure 5 illustrates the basic template for the artifact resolution profile.

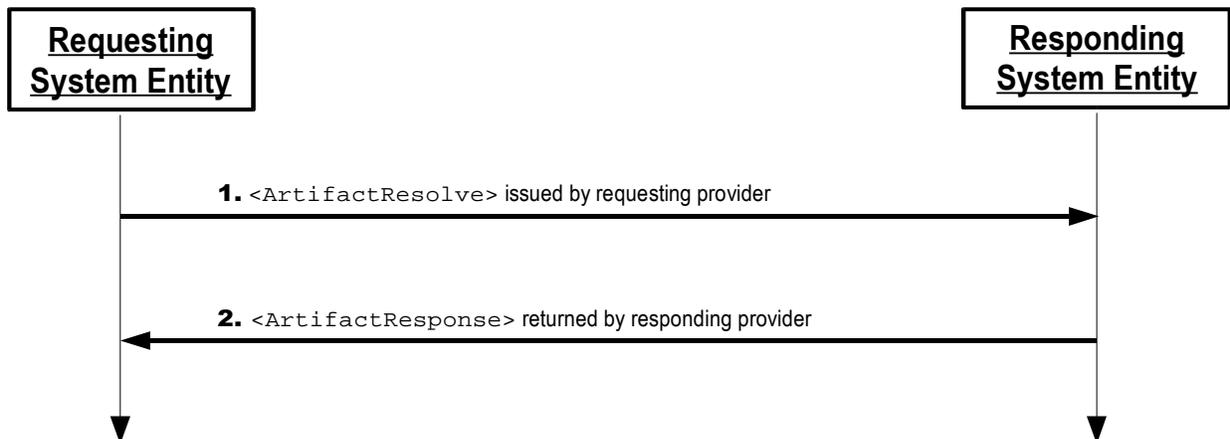


Figure 5

1410 The following steps are described by the profile.

#### 1411 1. <ArtifactResolve> issued by Requesting Entity

1412 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an  
1413 artifact issuer.

#### 1414 2. <ArtifactResponse> issued by Responding Entity

1415 In step 2, the responder (after processing the request) issues an <ArtifactResponse>  
1416 message to the requester.

## 1417 **5.3 Profile Description**

1418 In the descriptions below, the following is referred to:

### 1419 **Artifact Resolution Service**

1420 This is the artifact resolution protocol endpoint at an artifact issuer to which `<ArtifactResolve>`  
1421 messages are delivered.

### 1422 **5.3.1 `<ArtifactResolve>` issued by Requesting Entity**

1423 To initiate the profile, a requester, having received an artifact and determined the issuer using the  
1424 `SourceID`, sends an `<ArtifactResolve>` message containing the artifact to an artifact issuer's artifact  
1425 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this  
1426 endpoint and the bindings supported by the artifact issuer

1427 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the  
1428 request directly to the artifact issuer. The requester SHOULD authenticate itself to the identity provider,  
1429 either by signing the `<ArtifactResolve>` message or using any other binding-supported mechanism.  
1430 Specific profiles that use the HTTP Artifact binding MAY impose additional requirements such that  
1431 authentication is mandatory.

1432 Profile-specific rules for the contents of the `<ArtifactResolve>` message are included in Section 5.4.1.

### 1433 **5.3.2 `<ArtifactResponse>` issued by Responding Entity**

1434 The artifact issuer MUST process the `<ArtifactResolve>` message as defined in [SAMLCore]. After  
1435 processing the message or upon encountering an error, the artifact issuer MUST return an  
1436 `<ArtifactResponse>` message containing an appropriate status code to the requester to complete the  
1437 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the  
1438 artifact will also be included.

1439 The responder MUST authenticate itself to the requester, either by signing the `<ArtifactResponse>` or  
1440 using any other binding-supported mechanism.

1441 Profile-specific rules for the contents of the `<ArtifactResponse>` message are included in Section  
1442 5.4.2.

## 1443 **5.4 Use of Artifact Resolution Protocol**

### 1444 **5.4.1 `<ArtifactResolve>` Usage**

1445 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the requesting entity;  
1446 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1447 `format:entity`.

1448 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by  
1449 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact  
1450 binding MAY impose additional requirements such that authentication is mandatory.

### 1451 **5.4.2 `<ArtifactResponse>` Usage**

1452 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the artifact issuer;  
1453 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
1454 `format:entity`.

1455 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1456 the message or using a binding-specific mechanism.

## 1457 **5.5 Use of Metadata**

1458 [SAMLMeta] defines an indexed endpoint element, `<md:ArtifactResolutionService>`, to describe  
1459 supported bindings and location(s) to which a requester may send requests using this profile. The `index`  
1460 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's  
1461 `EndpointIndex` field.

1462

## 6 Assertion Query/Request Profile

1463 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis  
1464 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a  
1465 synchronous binding, such as the SOAP binding defined in [SAMLBind].

1466

### 6.1 Required Information

1467 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1468 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1469 **Description:** Given below.

1470 **Updates:** None.

1471

### 6.2 Profile Overview

1472 The message exchange and basic processing rules that govern this profile are largely defined by Section  
1473 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to  
1474 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to  
1475 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1476 Figure 6 illustrates the basic template for the query/request profile.

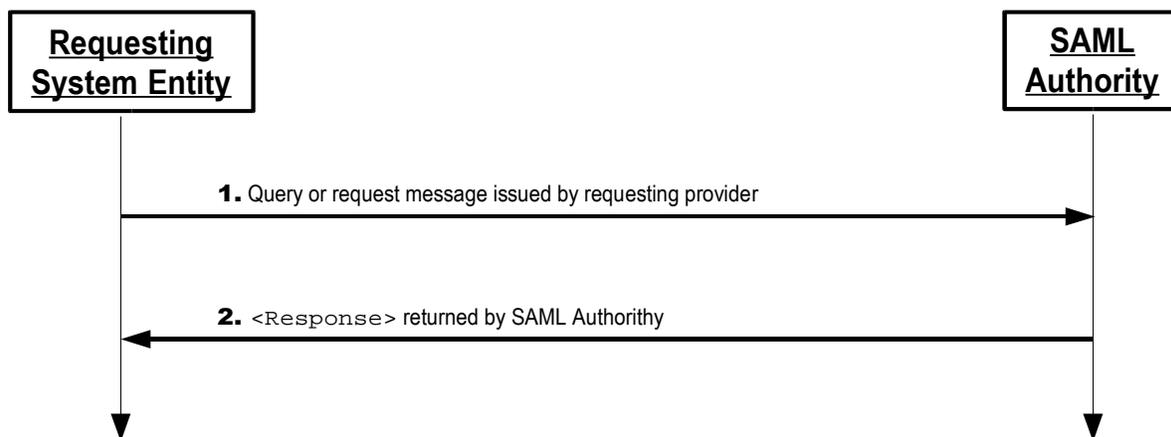


Figure 6

1477 The following steps are described by the profile.

#### 1478 1. Query/Request issued by Requesting Entity

1479 In step 1, a requester initiates the profile by sending an `<AssertionIDRequest>`,  
1480 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`  
1481 message to a SAML authority.

#### 1482 2. <Response> issued by SAML Authority

1483 In step 2, the responding SAML authority (after processing the query or request) issues a  
1484 `<Response>` message to the requester.

## 1485 **6.3 Profile Description**

1486 In the descriptions below, the following are referred to:

### 1487 **Query/Request Service**

1488 This is the query/request protocol endpoint at a SAML authority to which query or  
1489 <AssertionIDRequest> messages are delivered.

### 1490 **6.3.1 Query/Request issued by Requesting Entity**

1491 To initiate the profile, a requester issues an <AssertionIDRequest>, <SubjectQuery>,  
1492 <AuthnQuery>, <AttributeQuery>, or <AuthzDecisionQuery> message to a SAML authority's  
1493 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of  
1494 this endpoint and the bindings supported by the SAML authority.

1495 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the  
1496 request directly to the identity provider. The requester SHOULD authenticate itself to the SAML authority  
1497 either by signing the message or using any other binding-supported mechanism.

1498 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

### 1499 **6.3.2 <Response> issued by SAML Authority**

1500 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After  
1501 processing the message or upon encountering an error, the SAML authority MUST return a <Response>  
1502 message containing an appropriate status code to the requester to complete the SAML protocol  
1503 exchange. If the request is successful in locating one or more matching assertions, they will also be  
1504 included in the response.

1505 The responder SHOULD authenticate itself to the requester, either by signing the <Response> or using  
1506 any other binding-supported mechanism.

1507 Profile-specific rules for the contents of the <Response> message are included in Section 6.4.2.

## 1508 **6.4 Use of Query/Request Protocol**

### 1509 **6.4.1 Query/Request Usage**

1510 The <Issuer> element MUST be present.

1511 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by  
1512 signing the message or using a binding-specific mechanism.

### 1513 **6.4.2 <Response> Usage**

1514 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1515 SAML authority; the *Format* attribute MUST be omitted or have a value of  
1516 urn:oasis:names:tc:SAML:2.0:nameid-format:entity. Note that this need not necessarily match the  
1517 <Issuer> element in the returned assertion(s).

1518 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by  
1519 signing the message or using a binding-specific mechanism.

## 1520 **6.5 Use of Metadata**

1521 [SAMLMeta] defines several endpoint elements, <md:AssertionIDRequestService>,  
1522 <md:AuthnQueryService>, <md:AttributeService>, and <md:AuthzService>, to describe  
1523 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1524 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can  
1525 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an  
1526 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk  
1527 encryption key.

1528

## 7 Name Identifier Mapping Profile

1529 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a  
1530 different name identifier for the same principal. This profile describes the use of this protocol with a  
1531 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for  
1532 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

### 7.1 Required Information

1534 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1535 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1536 **Description:** Given below.

1537 **Updates:** None.

### 7.2 Profile Overview

1539 The message exchange and basic processing rules that govern this profile are largely defined by Section  
1540 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to  
1541 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to  
1542 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1543 Figure 7 illustrates the basic template for the name identifier mapping profile.

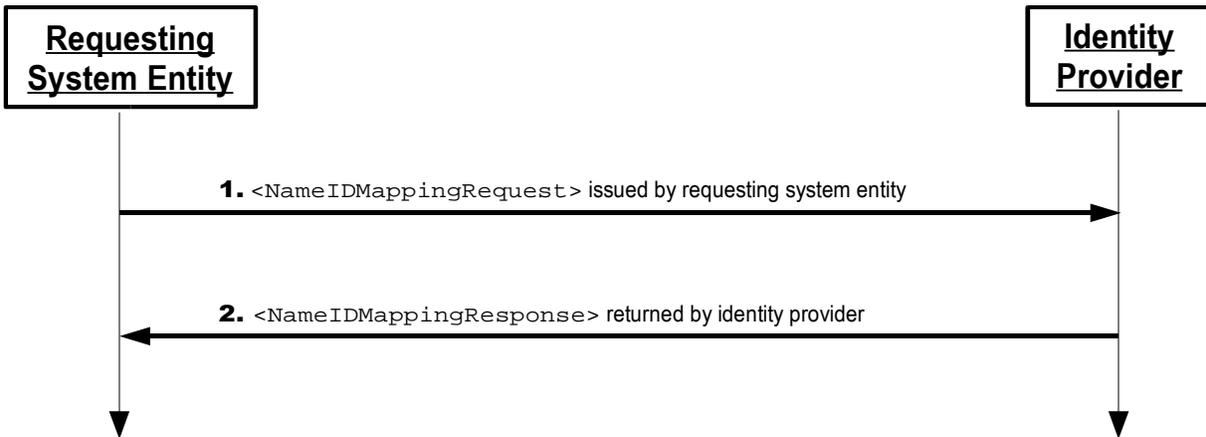


Figure 7

1544 The following steps are described by the profile.

#### 1545 1. <NameIDMappingRequest> issued by Requesting Entity

1546 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to  
1547 an identity provider.

#### 1548 2. <NameIDMappingResponse> issued by Identity Provider

1549 In step 2, the responding identity provider (after processing the request) issues a  
1550 <NameIDMappingResponse> message to the requester.

## 1551 **7.3 Profile Description**

1552 In the descriptions below, the following is referred to:

### 1553 **Name Identifier Mapping Service**

1554 This is the name identifier mapping protocol endpoint at an identity provider to which  
1555 <NameIDMappingRequest> messages are delivered.

### 1556 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1557 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's  
1558 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the  
1559 location of this endpoint and the bindings supported by the identity provider.

1560 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the  
1561 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,  
1562 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1563 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in  
1564 Section 7.4.1.

### 1565 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1566 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].  
1567 After processing the message or upon encountering an error, the identity provider MUST return a  
1568 <NameIDMappingResponse> message containing an appropriate status code to the requester to  
1569 complete the SAML protocol exchange.

1570 The responder MUST authenticate itself to the requester, either by signing the  
1571 <NameIDMappingResponse> or using any other binding-supported mechanism.

1572 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in  
1573 Section 7.4.2.

## 1574 **7.4 Use of Name Identifier Mapping Protocol**

### 1575 **7.4.1 <NameIDMappingRequest> Usage**

1576 The <Issuer> element MUST be present.

1577 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing  
1578 the message or using a binding-specific mechanism.

### 1579 **7.4.2 <NameIDMappingResponse> Usage**

1580 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding  
1581 identity provider; the `Format` attribute MUST be omitted or have a value of  
1582 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1583 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing  
1584 the message or using a binding-specific mechanism.

1585 Section 2.3.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In  
1586 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester  
1587 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it  
1588 in subsequent protocol messages or assertions.

#### 1589 **7.4.2.1 Limiting Use of Mapped Identifier**

1590 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning  
1591 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but  
1592 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>  
1593 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>  
1594 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying  
1595 parties, and MUST be signed for integrity protection.

#### 1596 **7.5 Use of Metadata**

1597 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported  
1598 bindings and location(s) to which a requester may send requests using this profile.

1599 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's  
1600 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate  
1601 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

---

## 1602 8 SAML Attribute Profiles

### 1603 8.1 Basic Attribute Profile

1604 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with  
1605 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas  
1606 to validate syntax.

#### 1607 8.1.1 Required Information

1608 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1609 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1610 **Description:** Given below.

1611 **Updates:** None.

#### 1612 8.1.2 SAML Attribute Naming

1613 The `NameFormat` XML attribute in `<Attribute>` elements MUST be  
1614 urn:oasis:names:tc:SAML:2.0:attrname-format:basic.

1615 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

##### 1616 8.1.2.1 Attribute Name Comparison

1617 Two `<Attribute>` elements refer to the same SAML attribute if and only if the values of their `Name` XML  
1618 attributes are equal in the sense of Section 3.3.6 of [Schema2].

#### 1619 8.1.3 Profile-Specific XML Attributes

1620 No additional XML attributes are defined for use with the `<Attribute>` element.

#### 1621 8.1.4 SAML Attribute Values

1622 The schema type of the contents of the `<AttributeValue>` element MUST be drawn from one of the  
1623 types defined in Section 3.3 of [Schema2]. The `xsi:type` attribute MUST be present and be given the  
1624 appropriate value.

#### 1625 8.1.5 Example

```
1626 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1627     Name="FirstName">  
1628     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1629 </saml:Attribute>
```

## 1630 8.2 X.500/LDAP Attribute Profile

1631 Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory  
1632 Access Protocol specifications [LDAP] are widely deployed. Organizations using these directories make  
1633 use of directory schema to model information to be stored in the directories. This includes common  
1634 schema defined in the X.500 and LDAP specifications themselves, schema defined in other public  
1635 documents (such as the Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson  
1636 schema [RFC2798]), and schema defined for particular private purposes. In any of these cases,  
1637 organizations may wish to reuse these schema definitions in the context of SAML attribute statements,

1638 and to do so in an interoperable fashion.  
1639 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such  
1640 attributes when expressed as SAML attributes.

## 1641 8.2.1 Required Information

1642 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP (this is also the target namespace  
1643 assigned in the corresponding X.500/LDAP profile schema document [SAMLLDAP-xsd])

1644 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1645 **Description:** Given below.

1646 **Updates:** None.

## 1647 8.2.2 SAML Attribute Naming

1648 The `NameFormat` XML attribute in `<Attribute>` elements MUST be  
1649 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1650 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.  
1651 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the  
1652 X.500/LDAP attribute type.

1653 Example:

```
1654 urn:oid:2.5.4.3
```

1655 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,  
1656 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1657 For purposes of human readability, there may also be a requirement for some applications to carry an  
1658 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in  
1659 [SAMLCore]) MAY be used for this purpose. If the definition of the X.500/LDAP attribute type includes one  
1660 or more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, SHOULD  
1661 be one of the defined descriptors.

### 1662 8.2.2.1 Attribute Name Comparison

1663 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1664 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the  
1665 comparison.

## 1666 8.2.3 Profile-Specific XML Attributes

1667 An additional XML attribute is defined in the XML namespace  
1668 urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP for use with the `<Attribute>` element:

1669 `Encoding` [Optional]

1670 The value of this XML attribute specifies the encoding used for the associated SAML attribute value.

1671 Only one value for this attribute is defined at this time: "LDAP". This specifies the use of the LDAP-specific  
1672 encoding for this X.500 attribute value, as described in Section 8.2.4. Future versions of this profile may  
1673 define additional encoding rules and will assign different values for this attribute.

## 1674 8.2.4 SAML Attribute Values

1675 X.500 attribute definitions for use in native X.500 directories specify the syntax of the attribute using  
1676 ASN.1 [ASN.1]. For transfer via the LDAP protocol, attribute definitions may additionally include an LDAP-  
1677 specific encoding, commonly of Unicode characters in UTF-8 form. This encoding is identified by an  
1678 LDAP-specific syntax. This SAML attribute profile only specifies the form of SAML attribute values for  
1679 those attributes for which an LDAP-specific syntax is provided. Future extensions to this profile may define

1680 attribute value formats for other X.500-defined attributes.

1681 For the following LDAP-specific syntaxes, the value of an X.500 attribute of this syntax is encoded as  
 1682 simply the UTF-8 string itself, as the content of the <AttributeValue> element, with no additional  
 1683 whitespace. In such cases, the `xsi:type` XML attribute MUST be set to **xs:string**. The profile-specific  
 1684 `Encoding` XML attribute is provided, with a value of "LDAP":

1685	Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1686	Bit String	1.3.6.1.4.1.1466.115.121.1.6
1687	Boolean	1.3.6.1.4.1.1466.115.121.1.7
1688	Country String	1.3.6.1.4.1.1466.115.121.1.11
1689	DN	1.3.6.1.4.1.1466.115.121.1.12
1690	Delivery Method	1.3.6.1.4.1.1466.115.121.1.14
1691	Directory String	1.3.6.1.4.1.1466.115.121.1.15
1692	DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16
1693	DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17
1694	Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21
1695	Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1696	Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1697	Guide	1.3.6.1.4.1.1466.115.121.1.25
1698	IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1699	INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1700	LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1701	Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1702	Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1703	Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1704	Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1705	Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1706	Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1707	Octet String	1.3.6.1.4.1.1466.115.121.1.40
1708	OID	1.3.6.1.4.1.1466.115.121.1.38
1709	Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1710	Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1711	Protocol Information	1.3.6.1.4.1.1466.115.121.1.42
1712	Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1713	Printable String	1.3.6.1.4.1.1466.115.121.1.44
1714	Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1715	Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1716	Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51
1717	Telex Number	1.3.6.1.4.1.1466.115.121.1.52
1718	UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1719 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>  
 1720 element, by base64-encoding [RFC2045] the encompassing ASN.1 OCTET STRING-encoded LDAP  
 1721 attribute value. The `xsi:type` XML attribute MUST be set to **xs:base64Binary**. The profile-specific  
 1722 `Encoding` XML attribute is provided, with a value of "LDAP".

1723 When comparing SAML attribute values for equality, the matching rules specified for the corresponding  
 1724 X.500/LDAP attribute type MUST be observed (case sensitivity, for example).

## 1725 8.2.5 Profile-Specific Schema

1726 The following schema defines the profile-specific `Encoding` XML attribute:

```

1727 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
1728         xmlns="http://www.w3.org/2001/XMLSchema"
1729         version="2.0">
1730   <attribute name="Encoding" type="string"/>
1731 </schema>

```

## 1732 8.2.6 Example

1733 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the  
1734 SAML assertion subject's first name. It's OID is 2.5.4.42 and the syntax is Directory String.

```
1735 <saml:Attribute  
1736 xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"  
1737 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1738 Name="urn:oid:2.5.4.42" FriendlyName="givenName"  
1739 ldapprof:Encoding="LDAP">  
1740 <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1741 </saml:Attribute>
```

## 1742 8.3 UUID Attribute Profile

1743 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and  
1744 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with  
1745 a UUID.

### 1746 8.3.1 Required Information

1747 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1748 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1749 **Description:** Given below.

1750 **Updates:** None.

### 1751 8.3.2 UUID and GUID Background

1752 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to  
1753 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs  
1754 were originally used in the Network Computing System (NCS), and then used in the Open Software  
1755 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in  
1756 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1757 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of  
1758 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,  
1759 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

```
1760 f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

1761 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a  
1762 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

### 1763 8.3.3 SAML Attribute Naming

1764 The NameFormat XML attribute in <Attribute> elements MUST be  
1765 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1766 If the underlying representation of the attribute's name is a UUID, then the URN uuid namespace  
1767 described in [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt>] is used. In this approach the  
1768 Name XML attribute is based on the URN form of the underlying UUID that identifies the attribute.

1769 **Example:**

```
1770 urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

1771 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used  
1772 in the Name XML attribute.

1773 For purposes of human readability, there may also be a requirement for some applications to carry an

1774 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in  
1775 [SAMLCORE]) MAY be used for this purpose.

### 1776 8.3.3.1 Attribute Name Comparison

1777 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1778 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The  
1779 `FriendlyName` attribute plays no role in the comparison.

### 1780 8.3.4 Profile-Specific XML Attributes

1781 No additional XML attributes are defined for use with the `<Attribute>` element.

### 1782 8.3.5 SAML Attribute Values

1783 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be  
1784 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST  
1785 be set to `xs:anyURI`.

1786 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`  
1787 element.

### 1788 8.3.6 Example

1789 The following is an example of a DCE Extended Registry Attribute, the "pre\_auth\_req" setting, which has a  
1790 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1791 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1792           Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"  
1793           FriendlyName="pre_auth_req">  
1794   <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>  
1795 </saml:Attribute>
```

## 1796 8.4 DCE PAC Attribute Profile

1797 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names  
1798 and values. It is used to standardize a mapping between the primary information that makes up a DCE  
1799 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in  
1800 Section 8.3.

### 1801 8.4.1 Required Information

1802 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace  
1803 assigned in the corresponding DCE PAC attribute profile schema document [SAMLDCExsd])

1804 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1805 **Description:** Given below.

1806 **Updates:** None.

### 1807 8.4.2 PAC Description

1808 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of  
1809 information is common across principals and makes up the bulk of a DCE identity:

- 1810 • The principal's DCE "realm" or "cell"
- 1811 • The principal's unique identifier
- 1812 • The principal's primary DCE local group membership

- 1813 • The principal's set of DCE local group memberships (multi-valued)
- 1814 • The principal's set of DCE foreign group memberships (multi-valued)

1815 The primary value(s) of each of these attributes is a UUID.

### 1816 **8.4.3 SAML Attribute Naming**

1817 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual  
1818 specific attribute names, rather than a naming convention.

1819 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST  
1820 be `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1821 For purposes of human readability, there may also be a requirement for some applications to carry an  
1822 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in  
1823 [SAMLCore]) MAY be used for this purpose.

1824 See Section 8.4.6 for the specific attribute names defined by this profile.

#### 1825 **8.4.3.1 Attribute Name Comparison**

1826 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1827 values are equal in the sense of [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt>]. The  
1828 `FriendlyName` attribute plays no role in the comparison.

### 1829 **8.4.4 Profile-Specific XML Attributes**

1830 No additional XML attributes are defined for use with the `<Attribute>` element.

### 1831 **8.4.5 SAML Attribute Values**

1832 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described  
1833 in Section 8.3.5 of the UUID profile is used to represent such values.

1834 However, additional information associated with the UUID value is permitted by this profile, consisting of a  
1835 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional  
1836 information is carried in the `<AttributeValue>` element in `FriendlyName` and `Realm` XML attributes  
1837 defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`. Note that this is not  
1838 the same as the `FriendlyName` XML attribute defined in [SAMLCore], although it has the same basic  
1839 purpose.

1840 The following schema defines the profile-specific XML attributes and a complex type used in an  
1841 `xsi:type` specification:

```
1842 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1843         xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1844         xmlns="http://www.w3.org/2001/XMLSchema"  
1845         version="2.0">  
1846     <attribute name="Realm" type="anyURI"/>  
1847     <attribute name="FriendlyName" type="string"/>  
1848     <complexType name="DCEValueType">  
1849         <simpleContent>  
1850             <extension base="anyURI">  
1851                 <attribute ref="dce:Realm" use="optional"/>  
1852                 <attribute ref="dce:FriendlyName" use="optional"/>  
1853             </extension>  
1854         </simpleContent>  
1855     </complexType>  
1856 </schema>
```

## 1857 8.4.6 Attribute Definitions

1858 The following are the set of SAML attributes defined by this profile. In each case, an `xsi:type` XML  
1859 attribute MAY be included in the `<AttributeValue>` element, but MUST have the value  
1860 `dce:DCEValueType`, where the `dce` prefix is arbitrary and MUST be bound to the XML namespace  
1861 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`.

1862 Note that such use of `xsi:type` will require validating attribute consumers to include the extension  
1863 schema defined by this profile.

### 1864 8.4.6.1 Realm

1865 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

1866 **Name:** `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm`

1867 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion  
1868 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the  
1869 realm's string name.

### 1870 8.4.6.2 Principal

1871 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

1872 **Name:** `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal`

1873 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion  
1874 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing  
1875 the principal's string name.

1876 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form  
1877 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section  
1878 8.4.6.1).

### 1879 8.4.6.3 Primary Group

1880 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

1881 **Name:** `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group`

1882 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion  
1883 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing  
1884 the group's string name.

1885 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form  
1886 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section  
1887 8.4.6.1).

### 1888 8.4.6.4 Groups

1889 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

1890 **Name:** `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups`

1891 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE group membership  
1892 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing  
1893 the group's string name.

1894 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form  
1895 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section  
1896 8.4.6.1).

#### 1897 8.4.6.5 Foreign Groups

1898 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

1899 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

1900 Each <AttributeValue> element contains a UUID in URN form identifying a DCE foreign group  
1901 membership of the SAML assertion subject, with an optional profile-specific FriendlyName XML attribute  
1902 containing the group's string name.

1903 The profile-specific Realm XML attribute MUST be included and MUST contain a UUID in URN form  
1904 identifying the DCE realm/cell of the foreign group.

#### 1905 8.4.7 Example

1906 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE  
1907 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local  
1908 groups and an "engineers" foreign group.

```
1909 <saml:Assertion
1910 xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE" ...>
1911   <saml:Issuer>...</saml:Issuer>
1912   <saml:Subject>...</saml:Subject>
1913   <saml:AttributeStatement>
1914     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1915       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">
1916       <saml:AttributeValue xsi:type="dce:DCEValueType"
1917 dce:FriendlyName="example.com">
1918         urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b
1919       </saml:AttributeValue>
1920     </saml:Attribute>
1921     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1922       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">
1923       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">
1924         urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b
1925       </saml:AttributeValue>
1926     </saml:Attribute>
1927     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1928       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">
1929       <saml:AttributeValue xsi:type="dce:DCEValueType"
1930 dce:FriendlyName="cubicle-dwellers">
1931         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
1932       </saml:AttributeValue>
1933     </saml:Attribute>
1934     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1935       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">
1936       <saml:AttributeValue xsi:type="dce:DCEValueType"
1937 dce:FriendlyName="cubicle-dwellers">
1938         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
1939       </saml:AttributeValue>
1940       <saml:AttributeValue xsi:type="dce:DCEValueType"
1941 dce:FriendlyName="underpaid">
1942         urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b
1943       </saml:AttributeValue>
1944     </saml:Attribute>
1945     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1946       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups">
1947       <saml:AttributeValue xsi:type="dce:DCEValueType"
1948 dce:FriendlyName="engineers"
1949 dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">
1950         urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b
1951       </saml:AttributeValue>
1952     </saml:Attribute>
1953   </saml:AttributeStatement>
1954 </saml:Assertion>
```

## 1955 **8.5 XACML Attribute Profile**

1956 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS  
1957 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute  
1958 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML  
1959 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute  
1960 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into  
1961 XACML attributes and used as input to XACML authorization decisions.

### 1962 **8.5.1 Required Information**

1963 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace  
1964 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

1965 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

1966 **Description:** Given below.

1967 **Updates:** None.

### 1968 **8.5.2 SAML Attribute Naming**

1969 The `NameFormat` XML attribute in `<Attribute>` elements MUST be  
1970 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1971 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1972 For purposes of human readability, there may also be a requirement for some applications to carry an  
1973 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in  
1974 [SAMLCore]) MAY be used for this purpose, but is not translatable into the XACML attribute equivalent.

#### 1975 **8.5.2.1 Attribute Name Comparison**

1976 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute  
1977 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

### 1978 **8.5.3 Profile-Specific XML Attributes**

1979 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-  
1980 valued XML attribute called `DataType` is defined in the XML namespace  
1981 urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML.

1982 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified  
1983 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

1984 While in principle any URI reference can be used as a data type, the standard values to be used are  
1985 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then  
1986 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values  
1987 must be extended to support the new data types.

### 1988 **8.5.4 SAML Attribute Values**

1989 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed  
1990 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data  
1991 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute  
1992 SHOULD also be used.

### 1993 **8.5.5 Profile-Specific Schema**

1994 The following schema defines the profile-specific `DataType` XML attribute:

```
1995 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
1996         xmlns="http://www.w3.org/2001/XMLSchema"
1997         version="2.0">
1998     <attribute name="DataType" type="anyURI"/>
1999 </schema>
```

## 2000 **8.5.6 Example**

2001 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the  
2002 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple  
2003 attribute profiles when they are compatible with each other.

```
2004 <saml:Attribute
2005     xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2006     xmlns:ldaprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2007     xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2008     ldaprof:Encoding="LDAP"
2009     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2010     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2011     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>
2012 </saml:Attribute>
```

---

## 9 References

2013

- 2014 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 2015 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,  
2016 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2017 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic  
2018 notation, ITU-T Recommendation X.680, July 2002. See  
2019 [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)  
2020 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2021 **[eduPerson]** eduPerson.Idif. See <http://www.educase.edu/eduperson>.
- 2022 **[LDAP]** J. Hodges et al., Lightweight Directory Access Protocol (v3): Technical Specification,  
2023 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2024 **[Mealling]** P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03.  
2025 January 2004
- 2026 **[MSURL]** Microsoft technical support article,  
2027 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2028 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See  
2029 [http://wp.netscape.com/newsref/std/cookie\\_spec.html](http://wp.netscape.com/newsref/std/cookie_spec.html).
- 2030 **[Rescorla-Sec]** E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations,  
2031 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2032 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 2033 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 2034 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 2035 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message  
2036 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 2037 **[RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC  
2038 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 2039 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 2040 **[RFC2256]** M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3,  
2041 December 1997
- 2042 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 2043 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 2044 **[RFC2617]** HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617,  
2045 <http://www.ietf.org/rfc/rfc2617.txt>.
- 2046 **[RFC2798]** M. Smith, Definition of the inetOrgPerson LDAP Object Class, IETF RFC 2798, April  
2047 200. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2048 **[RFC2965]** D. Cristol et al., HTTP State Management Mechanism, IETF RFC 2965, October 2000.  
2049 See <http://www.ietf.org/rfc/rfc2965.txt>.
- 2050 **[RFC3061]** M. Mealling, A URN Namespace of Object Identifiers, IETF RFC 3061, February 2001.  
2051 See <http://www.ietf.org/rfc/rfc3061.txt>.
- 2052 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML)*  
2053 *V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-bindings-2.0-cd-01. See  
2054 <http://www.oasis-open.org/committees/security/>.
- 2055 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup*  
2056 *Language (SAML) V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-core-2.0-  
2057 cd-01. See <http://www.oasis-open.org/committees/security/>.

- 2058 **[SAML DCE-xsd]** S. Cantor et al., SAML DCE PAC attribute profile schema. OASIS SSTC, August 2004.  
 2059 Document ID sstc-saml-schema-dce-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
 2060 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 2061 **[SAML ECP-xsd]** S. Cantor et al., SAML ECP profile schema. OASIS SSTC, August 2004. Document ID  
 2062 sstc-saml-schema-ecp-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2063 **[SAML Gloss]** J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*  
 2064 *V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-glossary-2.0-cd-01. See  
 2065 <http://www.oasis-open.org/committees/security/>.
- 2066 **[SAML LDAP-xsd]** S. Cantor et al., SAML LDAP attribute profile schema. OASIS SSTC, August 2004.  
 2067 Document ID sstc-saml-schema-ldap-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
 2068 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 2069 **[SAML Meta]** S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language (SAML)*  
 2070 *V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-metadata-2.0-cd-01. See  
 2071 <http://www.oasis-open.org/committees/security/>.
- 2072 **[SAML Reqs]** Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002,  
 2073 <http://www.oasis-open.org/committees/security/>.
- 2074 **[SAML Sec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security Assertion*  
 2075 *Markup Language (SAML) V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-  
 2076 sec-consider-2.0-cd-01. See <http://www.oasis-open.org/committees/security/>.
- 2077 **[SAML Web]** OASIS Security Services Technical Committee website, [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
 2078 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 2079 **[SAML XAC-xsd]** S. Cantor et al., SAML XACML attribute profile schema. OASIS SSTC, August 2004.  
 2080 Document ID sstc-saml-schema-xacml-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
 2081 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 2082 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium  
 2083 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>. Note that this  
 2084 specification normatively references [Schema2], listed below.
- 2085 **[Schema2]** Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes, W3C  
 2086 Recommendation 02 May 2001, <http://www.w3.org/TR/xmlschema-2/>
- 2087 **[SESSION]** RL “Bob” Morgan, Support of target web server sessions in Shibboleth,  
 2088 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>
- 2089 **[ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1,  
 2090 <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- 2091 **[SOAP1.1]** D. Box et al., Simple Object Access Protocol (SOAP) 1.1, World Wide Web  
 2092 Consortium Note, May 2000, <http://www.w3.org/TR/SOAP>.
- 2093 **[SSL3]** A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November  
 2094 1996.
- 2095 **[WEBSSO]** RL “Bob” Morgan, Interactions between Shibboleth and local-site web sign-on services,  
 2096 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt>
- 2097 **[X.500]** Information technology - Open Systems Interconnection - The Directory: Overview of  
 2098 concepts, models and services, ITU-T Recommendation X.500, February 2001. See  
 2099 [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.500)  
 2100 [X.500](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.500).
- 2101 **[XML Enc]** D. Eastlake et al., XML Encryption Syntax and Processing,  
 2102 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web  
 2103 Consortium.
- 2104 **[XML Sig]** D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web  
 2105 Consortium, <http://www.w3.org/TR/xmlsig-core/>.
- 2106 **[XACML]** T. Moses, ed., *OASIS eXtensible Access Control Markup Language (XACML)*  
 2107 *Versions 1.0, 1.1, and 2.0*. Available on the OASIS XACML TC web page at  
 2108 [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).

---

## 2109 Appendix A. Acknowledgments

2110 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
2111 Committee, whose voting members at the time of publication were:

- 2112 • Conor Cahill, AOL
- 2113 • Hal Lockhart, BEA Systems
- 2114 • Rick Randall, Booz Allen Hamilton
- 2115 • Ronald Jacobson, Computer Associates
- 2116 • Gavenraj Sodhi, Computer Associates
- 2117 • Tim Alsop, CyberSafe Limited
- 2118 • Paul Madsen, Entrust
- 2119 • Carolina Canales-Valenzuela, Ericsson
- 2120 • Dana Kaufman, Forum Systems
- 2121 • Irving Reid, Hewlett-Packard
- 2122 • Paula Austel, IBM
- 2123 • Maryann Hondo, IBM
- 2124 • Michael McIntosh, IBM
- 2125 • Anthony Nadalin, IBM
- 2126 • Nick Ragouzis, Individual
- 2127 • Scott Cantor, Internet2
- 2128 • Bob Morgan, Internet2
- 2129 • Prateek Mishra, Netegrity
- 2130 • Forest Yin, Netegrity
- 2131 • Peter Davis, Neustar
- 2132 • Frederick Hirsch, Nokia
- 2133 • John Kemp, Nokia
- 2134 • Senthil Sengodan, Nokia
- 2135 • Scott Kiestler, Novell
- 2136 • Steve Anderson, OpenNetwork
- 2137 • Ari Kermaier, Oracle
- 2138 • Vamsi Motukuru, Oracle
- 2139 • Darren Platt, Ping Identity
- 2140 • Jim Lien, RSA Security
- 2141 • John Linn, RSA Security
- 2142 • Rob Philpott, RSA Security
- 2143 • Dipak Chopra, SAP
- 2144 • Jahan Moreh, Sigaba
- 2145 • Bhavna Bhatnagar, Sun Microsystems
- 2146 • Jeff Hodges, Sun Microsystems
- 2147 • Eve Maler, Sun Microsystems
- 2148 • Ronald Monzillo, Sun Microsystems
- 2149 • Emily Xu, Sun Microsystems
- 2150 • Mike Beach, Boeing

- 2151 • Greg Whitehead, Trustgenix
- 2152 • James Vanderbeek, Vodafone

2153 The editors also would like to acknowledge the following people for their contributions to previous versions  
2154 of the OASIS Security Assertions Markup Language Standard:

- 2155 • Stephen Farrell, Baltimore Technologies
- 2156 • David Orchard, BEA Systems
- 2157 • Krishna Sankar, Cisco Systems
- 2158 • Zahid Ahmed, CommerceOne
- 2159 • Carlisle Adams, Entrust
- 2160 • Tim Moses, Entrust
- 2161 • Nigel Edwards, Hewlett-Packard
- 2162 • Joe Pato, Hewlett-Packard
- 2163 • Bob Blakley, IBM
- 2164 • Marlena Erdos, IBM
- 2165 • Marc Chanliau, Netegrity
- 2166 • Chris McLaren, Netegrity
- 2167 • Lynne Rosenthal, NIST
- 2168 • Mark Skall, NIST
- 2169 • Simon Godik, Overxeer
- 2170 • Charles Norwood, SAIC
- 2171 • Evan Prodromou, Securant
- 2172 • Robert Griffin, RSA Security (former editor)
- 2173 • Sai Allarvarpu, Sun Microsystems
- 2174 • Chris Ferris, Sun Microsystems
- 2175 • Emily Xu, Sun Microsystems
- 2176 • Mike Myers, Traceroute Security
- 2177 • Phillip Hallam-Baker, VeriSign (former editor)
- 2178 • James Vanderbeek, Vodafone
- 2179 • Mark O'Neill, Vordel
- 2180 • Tony Palmer, Vordel

2181 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
2182 input to the OASIS Security Assertions Markup Language specifications:

- 2183 • Thomas Gross, IBM
- 2184 • Birgit Pfitzmann, IBM

---

## Appendix B. Notices

2186 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
2187 might be claimed to pertain to the implementation or use of the technology described in this document or  
2188 the extent to which any license under such rights might or might not be available; neither does it represent  
2189 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
2190 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
2191 available for publication and any assurances of licenses to be made available, or the result of an attempt  
2192 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
2193 users of this specification, can be obtained from the OASIS Executive Director.

2194 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
2195 other proprietary rights which may cover technology that may be required to implement this specification.  
2196 Please address the information to the OASIS Executive Director.

2197 **Copyright © OASIS Open 2004. All Rights Reserved.**

2198 This document and translations of it may be copied and furnished to others, and derivative works that  
2199 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
2200 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
2201 this paragraph are included on all such copies and derivative works. However, this document itself may  
2202 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
2203 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
2204 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
2205 into languages other than English.

2206 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
2207 or assigns.

2208 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2209 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
2210 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
2211 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.