



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

Last-Call Working Draft 15, 13 July 2004

Document identifier:

sstc-saml-profiles-2.0-draft-15

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:

Frederick Hirsch, Nokia
Scott Cantor, Internet2
John Hughes, Entegriy
Prateek Mishra, Netegrity
Eve Maler, Sun Microsystems

Contributors:

TBD

Abstract:

This specification defines profiles for the use of SAML assertions and request-response messages in communications protocols and frameworks, as well as attribute syntax for use in attribute statements.

Status:

This is a last-call working draft produced by the Security Services Technical Committee. **See the Revision History for details of changes made in this revision.**

Comments on this last-call draft are solicited by **2 August 2004** so that the TC can subsequently prepare an OASIS Committee Draft. Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them by filling in the form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security. The committee will publish vetted errata on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

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

Table of Contents

35		
36	1 Introduction.....	6
37	1.1 Profile Concepts.....	6
38	2 Specification of Additional Profiles.....	8
39	2.1 Guidelines for Specifying Profiles.....	8
40	2.2 Guidelines for Specifying Attribute Profiles.....	8
41	3 Confirmation Method Identifiers.....	10
42	3.1 Holder of Key.....	10
43	3.2 Sender Vouches.....	10
44	3.3 Bearer.....	11
45	4 SSO Profiles of SAML.....	12
46	4.1 Web Browser SSO Profile.....	12
47	4.1.1 Required Information.....	12
48	4.1.2 Profile Overview.....	12
49	4.1.3 Profile Description.....	13
50	4.1.3.1 HTTP Request to Service Provider.....	13
51	4.1.3.2 Service Provider Determines Identity Provider.....	14
52	4.1.3.3 <AuthnRequest> issued by Service Provider to Identity Provider.....	14
53	4.1.3.4 Identity Provider identifies Principal.....	14
54	4.1.3.5 Identity Provider issues <Response> to Service Provider.....	14
55	4.1.3.6 Service Provider grants or denies access to User Agent.....	15
56	4.1.4 Use of Authentication Request Protocol.....	15
57	4.1.4.1 <AuthnRequest> Usage.....	15
58	4.1.4.2 <Response> Usage.....	15
59	4.1.4.3 <Response> Message Processing Rules.....	16
60	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	17
61	4.1.4.5 POST-Specific Processing Rules.....	17
62	4.1.5 Unsolicited Responses.....	17
63	4.1.6 Use of Metadata.....	17
64	4.2 Enhanced Client and Proxy (ECP) Profile.....	18
65	4.2.1 Required Information.....	18
66	4.2.2 Preliminaries.....	18
67	4.2.3 Step 1: Accessing the Service Provider: ECP>SP.....	20
68	4.2.4 Steps 2,3: SOAP Message containing <AuthnRequest>: SP>ECP>IDP.....	20
69	4.2.4.1 PAOS Request Header Block: SP>ECP.....	21
70	4.2.4.2 ECP Request Header Block : SP > ECP.....	21
71	4.2.4.3 ECP RelayState Header Block : SP > ECP.....	22
72	4.2.4.4 SP>ECP Request Example.....	22
73	4.2.4.5 ECP>IDP Request Example.....	23
74	4.2.5 Steps 4,5: Authentication Response SOAP Message: IDP>ECP>SP.....	23
75	4.2.5.1 ECP Response Header Block : IDP > ECP.....	24
76	4.2.5.2 IDP>ECP Response Example.....	24
77	4.2.5.3 PAOS Response Header Block : ECP>SP.....	25
78	4.2.5.4 ECP>SP Response Example.....	25

79	4.2.6 Step 6: HTTP service response: SP>ECP.....	25
80	4.2.7 Security Considerations.....	26
81	4.2.8 ECP Profile XML Schema.....	26
82	4.3 Identity Provider Discovery Profile.....	27
83	4.3.1 Common Domain Cookie.....	27
84	4.3.2 Setting the Common Domain Cookie.....	27
85	4.3.3 Obtaining the Common Domain Cookie.....	28
86	4.4 Single Logout Profile.....	28
87	4.4.1 Required Information.....	28
88	4.4.2 Profile Overview.....	28
89	4.4.3 Profile Description.....	29
90	4.4.3.1 <LogoutRequest> issued by Service Provider to Identity Provider.....	29
91	4.4.3.2 Identity Provider determines Session Participants.....	30
92	4.4.3.3 <LogoutRequest> issued by Identity Provider to Session Participant/Authority.....	30
93	4.4.3.4 Session Participant/Authority issues <LogoutResponse> to Identity Provider.....	30
94	4.4.3.5 Identity Provider issues <LogoutResponse> to Service Provider.....	31
95	4.4.4 Use of Single Logout Protocol.....	31
96	4.4.4.1 <LogoutRequest> Usage.....	31
97	4.4.4.2 <LogoutResponse> Usage.....	32
98	4.4.5 Use of Metadata.....	32
99	4.5 Name Identifier Management Profile.....	32
100	4.5.1 Required Information.....	32
101	4.5.2 Profile Overview.....	32
102	4.5.3 Profile Description.....	33
103	4.5.3.1 <ManageNameIDRequest> issued by Requesting Identity/Service Provider.....	33
104	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	34
105	4.5.4 Use of Name Identifier Management Protocol.....	34
106	4.5.4.1 <ManageNameIDRequest> Usage.....	34
107	4.5.4.2 <ManageNameIDResponse> Usage.....	34
108	4.5.5 Use of Metadata.....	35
109	5 Artifact Resolution Profile.....	36
110	5.1 Required Information.....	36
111	5.2 Profile Overview.....	36
112	5.3 Profile Description.....	36
113	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	36
114	5.3.2 <ArtifactResponse> issued by Responding Entity.....	37
115	5.4 Use of Artifact Resolution Protocol.....	37
116	5.4.1 <ArtifactResolve> Usage.....	37
117	5.4.2 <ArtifactResponse> Usage.....	37
118	5.5 Use of Metadata.....	37
119	6 Assertion Query/Request Profile.....	38
120	6.1 Required Information.....	38
121	6.2 Profile Overview.....	38
122	6.3 Profile Description.....	38
123	6.3.1 Query/Request issued by Requesting Entity.....	38

124	6.3.2 <Response> issued by SAML Authority.....	39
125	6.4 Use of Query/Request Protocol.....	39
126	6.4.1 Query/Request Usage.....	39
127	6.4.2 <Response> Usage.....	39
128	6.5 Use of Metadata.....	39
129	7 Name Identifier Mapping Profile.....	40
130	7.1 Required Information.....	40
131	7.2 Profile Overview.....	40
132	7.3 Profile Description.....	40
133	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	40
134	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	41
135	7.4 Use of Name Identifier Mapping Protocol.....	41
136	7.4.1 <NameIDMappingRequest> Usage.....	41
137	7.4.2 <NameIDMappingResponse> Usage.....	41
138	7.4.2.1 Limiting Use of Mapped Identifier.....	41
139	7.5 Use of Metadata.....	41
140	8 SAML Attribute Profiles.....	43
141	8.1 Basic Attribute Profile.....	43
142	8.1.1 Required Information.....	43
143	8.1.2 SAML Attribute Naming.....	43
144	8.1.3 Profile-Specific XML Attributes.....	43
145	8.1.4 <AttributeDesignator> Comparison.....	43
146	8.1.5 SAML Attribute Values.....	43
147	8.1.6 Example.....	43
148	8.2 X.500/LDAP Attribute Profile.....	43
149	8.2.1 Required Information.....	44
150	8.2.2 SAML Attribute Naming.....	44
151	8.2.3 Profile-Specific XML Attributes.....	44
152	8.2.4 <AttributeDesignator> Comparison.....	44
153	8.2.5 SAML Attribute Values.....	44
154	8.2.6 Example.....	45
155	8.3 UUID Attribute Profile.....	45
156	8.3.1 Required Information.....	45
157	8.3.2 UUID and GUID Background.....	45
158	8.3.3 SAML Attribute Naming.....	45
159	8.3.4 Profile-Specific XML Attributes.....	45
160	8.3.5 <AttributeDesignator> Comparison.....	46
161	8.3.6 SAML Attribute Values.....	46
162	8.3.7 Standard DCE Attributes.....	46
163	8.3.7.1 Principal.....	46
164	8.3.7.2 Primary Group.....	46
165	8.3.7.3 Groups.....	46
166	8.3.8 Example.....	46
167	8.4 XACML Attribute Profile.....	46

168 8.4.1 Required Information.....47
169 8.4.2 SAML Attribute Naming.....47
170 8.4.3 Profile-Specific XML Attributes.....47
171 8.4.4 <AttributeDesignator> Comparison.....47
172 8.4.5 SAML Attribute Values.....47
173 8.4.6 Profile-Specific Schema.....47
174 8.4.7 Example.....48
175 9 References.....49

1 Introduction

176

177 This document specifies profiles for the use of SAML assertions and request-response messages in
178 communications protocols and frameworks, as well as attribute syntax for use in attribute statements.

179 A separate specification [SAMLCore] defines the SAML assertions and request-response protocol
180 messages themselves and another [SAMLBind] defines bindings of SAML messages to underlying
181 communications and messaging protocols.

1.1 Profile Concepts

182

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

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

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

197 A particular example of the latter are those that address SAML attributes. The SAML <Attribute> (and
198 <AttributeDesignator>) elements provide a great deal of flexibility in attribute naming, value syntax,
199 and including in-band metadata through the use of XML attributes. Interoperability is achieved by
200 constraining this flexibility when warranted by adhering to profiles that define how to use these elements
201 with greater specificity than the generic rules defined by [SAMLCore].

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

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

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

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

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

212 `Example code listings appear like this.`

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

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

- 216 • The prefix `saml`: stands for the SAML assertion namespace [SAMLCore].

- 217 • The prefix `samlp:` stands for the SAML request-response protocol namespace [SAMLCore].
 - 218 • The prefix `md:` stands for the SAML metadata namespace [SAMLMeta].
 - 219 • The prefix `ds:` stands for the W3C XML Signature namespace,
220 <http://www.w3.org/2000/09/xmldsig#> [XMLSig].
 - 221 • The prefix `xenc:` stands for the W3C XML Encryption namespace,
222 <http://www.w3.org/2001/04/xmlenc#>.
 - 223 • The prefix `SOAP-ENV:` stands for the SOAP 1.1 namespace,
224 <http://schemas.xmlsoap.org/soap/envelope> [SOAP1.1].
- 225 This specification uses the following typographical conventions in text: `<SAMLElement>`,
226 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`. In some cases, angle brackets are used
227 to indicate non-terminals, rather than XML elements; the intent will be clear from the context.

228

2 Specification of Additional Profiles

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

233 The SSTC welcomes submission of proposals from OASIS members for new profiles. OASIS members
234 may wish to submit these proposals for consideration by the SSTC in a future version of this specification.
235 Other members may simply wish to inform the committee of their work related to SAML. Please refer to
236 the SSTC web site for further details on how to submit such proposals to the SSTC.

2.1 Guidelines for Specifying Profiles

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

- 239 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
240 author, and provide reference to previously defined profiles that the new profile updates or
241 obsoletes.
- 242 2. Describe the set of interactions between parties involved in the profile. Any restrictions on
243 applications used by each party and the protocols involved in each interaction must be explicitly
244 called out.
- 245 3. Identify the parties involved in each interaction, including how many parties are involved and
246 whether intermediaries may be involved.
- 247 4. Specify the method of authentication of parties involved in each interaction, including whether
248 authentication is required and acceptable authentication types.
- 249 5. Identify the level of support for message integrity, including the mechanisms used to ensure
250 message integrity.
- 251 6. Identify the level of support for confidentiality, including whether a third party may view the contents
252 of SAML messages and assertions, whether the profile requires confidentiality, and the
253 mechanisms recommended for achieving confidentiality.
- 254 7. Identify the error states, including the error states at each participant, especially those that receive
255 and process SAML assertions or messages.
- 256 8. Identify security considerations, including analysis of threats and description of countermeasures.
- 257 9. Identify SAML confirmation method identifiers defined and/or utilized by the profile.
- 258 10. Identify relevant SAML metadata defined and/or utilized by the profile.

2.2 Guidelines for Specifying Attribute Profiles

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

- 261 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
262 author, and provide reference to previously defined profiles that the new profile updates or
263 obsoletes.
- 264 2. Syntax and restrictions on the acceptable values of the `NameFormat` and `Name` attributes of SAML
265 `<AttributeDesignator>` and `<Attribute>` elements.
- 266 3. Any additional namespace-qualified XML attributes defined by the profile that may be used in SAML
267 `<AttributeDesignator>` and `<Attribute>` elements.

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

272

3 Confirmation Method Identifiers

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

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

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

3.1 Holder of Key

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

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

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

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

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

```
300 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">  
301   <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
302     <ds:KeyInfo>  
303       <ds:KeyName>By-Tor</ds:KeyName>  
304     </ds:KeyInfo>  
305     <ds:KeyInfo>  
306       <ds:KeyName>Snow Dog</ds:KeyName>  
307     </ds:KeyInfo>  
308   </SubjectConfirmationData>  
309 </SubjectConfirmation>
```

3.2 Sender Vouches

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

312 Indicates that no other information is available about the context of use of the assertion. The relying party

313 SHOULD utilize other means to determine if it should process the assertion further, subject to optional
314 constraints on confirmation using the attributes that MAY be present in the
315 <SubjectConfirmationData> element, as defined by [SAMLCore].

316 3.3 Bearer

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

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

321 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
322 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
323 19th, 2004, in response to a request with ID "_1234567890".

```
324 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
325   <SubjectConfirmationData InResponseTo="_1234567890"  
326     Recipient="https://www.serviceprovider.com/saml/consumer"  
327     NotOnOrAfter="2004-03-19T13:27:00Z"  
328   </SubjectConfirmationData>  
329 </SubjectConfirmation>
```

330 4 SSO Profiles of SAML

331 A set of profiles are defined to support single sign-on of browsers and other client devices.

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

338 4.1 Web Browser SSO Profile

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

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

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

350 4.1.1 Required Information

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

352 **Contact information:** security-services-comment@lists.oasis-open.org

353 **SAML Confirmation Method Identifiers:** The SAML 2.0 "bearer" confirmation method identifier is used
354 by this profile. The following RECOMMENDED identifier has been assigned to this confirmation method:

355 urn:oasis:names:tc:SAML:2.0:cm:bearer

356 **Description:** Given below.

357 **Updates:** SAML 1.1 browser artifact and POST profiles and bearer confirmation method.

358 4.1.2 Profile Overview

359 The following figure illustrates the basic template for achieving SSO:

360 <need figure>

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

364 1. HTTP Request to Service Provider

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

367 **2. Service Provider Determines Identity Provider**

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

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

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

376 **4. Identity Provider identifies Principal**

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

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

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

386 **6. Service Provider grants or denies access to Principal**

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

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

392 **4.1.3 Profile Description**

393 If the profile is initiated by the service provider, start with section 4.1.3.1. If initiated by the identity provider,
394 start with section 4.1.3.5. In the descriptions below, the following are referred to:

395 **Single Sign-On Service**

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

398 **Assertion Consumer Service**

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

401 **4.1.3.1 HTTP Request to Service Provider**

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

408 **4.1.3.2 Service Provider Determines Identity Provider**

409 This step is implementation-dependent. The service provider MAY use the SAML identity provider
410 discovery profile, described in section 4.3. The service provider MAY also choose to redirect the user
411 agent to another service that is able to determine an appropriate identity provider. In such a case, the
412 service provider may issue an `<AuthnRequest>` (as in the next step) to this service to be relayed to the
413 identity provider, or it may rely on the intermediary service to issue an `<AuthnRequest>` message on its
414 behalf.

415 **4.1.3.3 `<AuthnRequest>` issued by Service Provider to Identity Provider**

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

421 The exact format of this HTTP response and the subsequent HTTP request to the single sign-on service
422 is defined by the SAML binding used. Profile-specific rules for the contents of the `<AuthnRequest>`
423 message are included in section 4.1.4.1. If the HTTP Redirect or POST binding is used, the
424 `<AuthnRequest>` message is delivered directly to the identity provider in this step. If the HTTP Artifact
425 binding is used, the Artifact Resolution profile defined in section 5 is used by the identity provider, which
426 makes a callback to the service provider to retrieve the `<AuthnRequest>` message, using for example
427 the SOAP binding.

428 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or TLS
429 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The `<AuthnRequest>` message MAY
430 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
431 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

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

435 **4.1.3.4 Identity Provider identifies Principal**

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

443 **4.1.3.5 Identity Provider issues `<Response>` to Service Provider**

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

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

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

457 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 ([SSL3]) or TLS
458 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <Assertion> element(s) in the
459 <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-
460 Artifact binding is used.

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

463 **4.1.3.6 Service Provider grants or denies access to User Agent**

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

469 **4.1.4 Use of Authentication Request Protocol**

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

474 **4.1.4.1 <AuthnRequest> Usage**

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

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

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

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

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

493 **4.1.4.2 <Response> Usage**

494 If the identity provider wishes to return an error, it MUST NOT include any assertions in the <Response>
495 message. Otherwise, if the request is successful (or if the response is not associated with a request), the

496 <Response> element MUST conform to the following:

- 497 • The <Issuer> element MAY be omitted, but if present it MUST contain the unique identifier of
498 the issuing identity provider; the Format attribute MUST be omitted or have a value of
499 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 500 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST
501 contain the unique identifier of the issuing identity provider; the Format attribute MUST be
502 omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 503 • The set of one or more assertions MUST contain at least one <AuthnStatement> that
504 reflects the authentication of the principal to the identity provider.
- 505 • At least one assertion containing an <AuthnStatement> MUST contain a <Subject>
506 element with at least one <SubjectConfirmation> element containing a Method of
507 urn:oasis:names:tc:SAML:2.0:cm:bearer. If the identity provider supports the Single
508 Logout profile, defined in section 4.4, any such authentication statements MUST include a
509 SessionIndex attribute to enable per-session logout requests by the service provider.
- 510 • Any bearer <SubjectConfirmationData> elements MUST contain a Recipient attribute
511 containing the service provider's assertion consumer service URL and a NotOnOrAfter
512 attribute that limits the window during which the assertion can be delivered. It MAY contain an
513 Address attribute limiting the client address from which the assertion can be delivered. It
514 MUST NOT contain a NotBefore attribute. If the containing message is in response to an
515 <AuthnRequest>, then the InResponseTo attribute MUST match the request's ID.
- 516 • Other statements and confirmation methods MAY be included in the assertion(s) at the
517 discretion of the identity provider. In particular, <AttributeStatement> elements MAY be
518 included. The <AuthnRequest> MAY contain an AttributeConsumingServiceIndex
519 XML attribute referencing information about desired or required attributes in [SAMLMeta]. The
520 identity provider MAY ignore this, or send other attributes at its discretion.
- 521 • The assertion(s) containing a bearer subject confirmation MUST contain an
522 <AudienceRestriction> including the service provider's unique identifier as an
523 <Audience>.
- 524 • Other conditions (and other <Audience> elements) MAY be included as requested by the
525 service provider or at the discretion of the identity provider. (Of course, any such conditions
526 MUST be understood by and accepted by the service provider in order for the assertion to be
527 considered valid.) The identity provider is NOT obligated to honor the requested set of
528 <Conditions> in the <AuthnRequest>, if any.

529 4.1.4.3 <Response> Message Processing Rules

530 Regardless of the SAML binding used, the service provider MUST:

- 531 • verify any signatures present on the assertion(s) or the response
- 532 • verify that the Recipient attribute in any bearer <SubjectConfirmationData> matches
533 the assertion consumer service URL to which the <Response> or artifact was delivered
- 534 • verify that the NotOnOrAfter attribute in any bearer <SubjectConfirmationData> has
535 not passed, subject to allowable clock skew between the providers
- 536 • verify that the InResponseTo attribute in the bearer <SubjectConfirmationData> equals
537 the ID of its original <AuthnRequest> message, unless the response is unsolicited (see
538 section 4.5) in which case the attribute MUST NOT be present
- 539 • verify that any assertions relied upon are valid in other respects

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

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

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

547 **4.1.4.4 Artifact-Specific `<Response>` Message Processing Rules**

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

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

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

554 **4.1.4.5 POST-Specific Processing Rules**

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

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

560 **4.1.5 Unsolicited Responses**

561 An identity provider may initiate this profile by delivering an unsolicited `<Response>` message to a service
562 provider.

563 An unsolicited `<Response>` MUST NOT contain an `InResponseTo` attribute, nor should any bearer
564 `<SubjectConfirmationData>` elements. If metadata as in [SAMLMeta] is used, the `<Response>` or
565 artifact SHOULD be delivered to the `<md:AssertionConsumerService>` endpoint of the service
566 provider labeled with the `isDefault` attribute.

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

570 **4.1.6 Use of Metadata**

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

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

578 The providers MAY document the key(s) used to sign requests, responses, and assertions with
579 `<md:KeyDescriptor>` elements with a `use` attribute of `sign`. When encrypting SAML elements,

580 <md:KeyDescriptor> elements with a use attribute of encrypt MAY be used to document supported
581 encryption algorithms and settings, and public keys used to receive bulk encryption keys.

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

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

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

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

598 4.2 Enhanced Client and Proxy (ECP) Profile

599 In the scenario supported by the enhanced client and proxy profile, a user of an enhanced client or proxy
600 either accesses a resource at a service provider, or accesses an identity provider such that the service
601 provider and desired resource are understood or implicit. The user authenticates (or has already
602 authenticated) to the identity provider, which then produces an authentication assertion (possibly with input
603 from the service provider) and the service provider consumes the assertion to establish a security context
604 for the user. During this process, a name identifier might also be established between the providers for the
605 principal, subject to the parameters of the interaction and the consent of the parties.

606 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction
607 with the Reverse-SOAP binding.

608 It is assumed that the user is using an enhanced client or proxy (see below) and can authenticate to the
609 identity provider by some means outside the scope of SAML.

610 4.2.1 Required Information

611 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp

612 **Contact information:** security-services-comment@lists.oasis-open.org

613 **SAML Confirmation Method Identifiers:** The SAML 2.0 "bearer" confirmation method identifier is used
614 by this profile. The following RECOMMENDED identifier has been assigned to this confirmation method:

615 urn:oasis:names:tc:SAML:2.0:cm:bearer

616 **Description:** Given below.

617 **Updates:** None.

618 4.2.2 Preliminaries

619 The Enhanced Client and Proxy (ECP) profile specifies interactions between enhanced clients and/or
620 proxies, service providers, and identity providers. It is a generalization of the browser profile described in
621 section 4.1, and makes reference to it in a number of respects. If not otherwise specified by this profile

622 (and if not specific to the use of browser-based bindings), the rules specified in section 4.1 MUST be
623 observed.

624 An enhanced client or proxy (ECP) is a client or proxy that:

625 1. Has, or knows how to obtain, knowledge about the identity provider that the principal associated
626 with the client wishes to use with the service provider.

627 • This allows a service provider to make an authentication request to such a client without the
628 need to know or discover the appropriate identity provider (effectively bypassing step 2 of the
629 browser profile).

630 2. Is able to use a reverse SOAP (PAOS) binding as profiled here for an authentication request and
631 response.

632 • This enables a service provider to obtain an authentication assertion from a client that is not
633 necessarily directly addressable and not necessarily continuously available.

634 • It leverages the benefits of SOAP while using a well-defined exchange pattern and profile to
635 enable interoperability.

636 • The enhanced client may be viewed as a SOAP intermediary between the service provider and
637 the identity provider.

638 The enhanced client may be a browser or some other user agent that supports the functionality described
639 in this profile. An enhanced proxy is an HTTP proxy (typically a WAP gateway) that emulates an enhanced
640 client. Unless stated otherwise, all statements referring to enhanced clients are to be understood as
641 statements about both enhanced clients as well as enhanced client proxies.

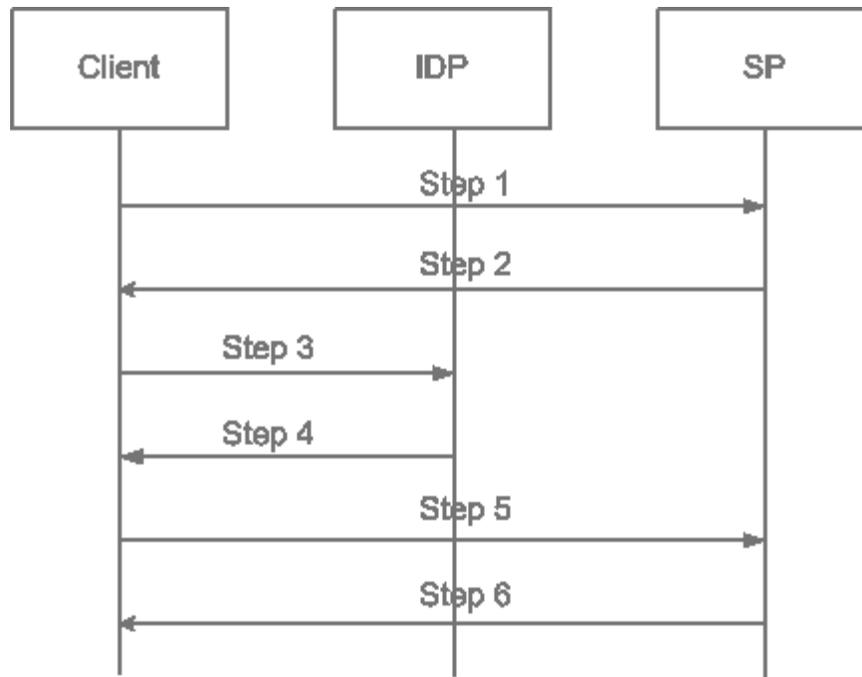
642 Since the enhanced client sends and receives messages in the body of HTTP requests and responses, it
643 has no arbitrary restrictions on the size of the protocol messages.

644 This profile leverages the Reverse SOAP binding [SAMLBind]. Implementers of this profile MUST follow
645 the rules for HTTP indications of PAOS support specified in that binding, in addition to those specified in
646 this profile. This specification profiles a PAOS SOAP header block conveyed between the HTTP
647 responder and the ECP but does not define PAOS. The PAOS specification is normative in case of
648 question regarding PAOS PAOS.

649 This profile defines SOAP header blocks that accompany the SAML requests and responses. These
650 header blocks may be composed with other SOAP header blocks as necessary, for example with the
651 SOAP Message Security WSS header block to add security features if needed, for example encryption of
652 the authentication request.

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

656 The following diagram shows the processing flow in the ECP profile:



657 **4.2.3 Step 1: Accessing the Service Provider: ECP>SP**

658 In step 1, the ECP accesses the service provider with an HTTP request. This HTTP request MUST
 659 conform to the PAOS binding, which means it must include the following HTTP header fields:

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

667 To give an example, a user-agent may request a page from the SP as follows:

```

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

672 **4.2.4 Steps 2,3: SOAP Message containing <AuthnRequest>: SP>ECP>IDP**

673 When the service provider requires a security context for the principal before providing a service or data, it
 674 can respond to the HTTP request using the PAOS binding with an <AuthnRequest> message in the
 675 HTTP response. The service provider will issue an HTTP 200 OK response to the ECP containing a single
 676 SOAP envelope.

677 The SOAP envelope MUST contain:

- 678 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the
 679 identity provider.
- 680 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of
 681 http://schemas.xmlsoap.org/soap/actor/next/. This header block provides control
 682 information such as the URL to which to send the response in this solicit-response message

683 exchange pattern.

684 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
685 `http://schemas.xmlsoap.org/soap/actor/next/`. The ECP Request header block defines
686 information related to the authentication request that the ECP may need to process it, such as a list
687 of identity providers acceptable to the service provider, whether the ECP may interact with the
688 principal through the client, and the service provider's human-readable name that may be displayed
689 to the principal.

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

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

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

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

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

700 **4.2.4.1 PAOS Request Header Block: SP>ECP**

701 The PAOS Request header block signals the use of PAOS processing and includes the following
702 attributes:

703 `responseConsumerURL` [Required]

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

709 `service` [Required]

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

712 `S:mustUnderstand` [Required]

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

715 `S:actor` [Required]

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

717 `messageID` [Optional]

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

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

722 **4.2.4.2 ECP Request Header Block : SP > ECP**

723 The ECP Request SOAP header block is used to convey information needed by the ECP to process the

724 authentication request. It is mandatory and its presence signals the use of this profile. It contains the
725 following elements and attributes:

726 `S:mustUnderstand` [Required]

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

729 `S:actor` [Required]

730 The value MUST be <http://schemas.xmlsoap.org/soap/actor/next/>.

731 `ProviderName` [Optional]

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

733 `IsPassive` [Optional]

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

737 `<saml:Issuer>` [Required]

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

741 `<samlp:IDPList>` [Optional]

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

744 See section 4.2.8 for the XML schema that defines this header block.

745 **4.2.4.3 ECP RelayState Header Block : SP > ECP**

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

749 `S:mustUnderstand` [Required]

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

751 `S:actor` [Required]

752 The value MUST be <http://schemas.xmlsoap.org/soap/actor/next/>.

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

758 See section 4.2.8 for the XML schema that defines this header block.

759 **4.2.4.4 SP>ECP Request Example**

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

```
761 <S:Envelope  
762     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
763     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
764     xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
```

```

765 <S:Header>
766   <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
767     responseConsumerURL="http://identity-service.example.com/abc"
768     messageID="6c3a4f8b9c2d" S:actor="next" S:mustUnderstand="1"
769     service="urn:oasis:names:tc:SAML:2.0:profiles:ecp">
770   </paos:Request>
771   <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
772     S:mustUnderstand="1" S:actor="http://schemas.xmlsoap.org/soap/actor/next/"
773     ProviderName="Service Provider X" IsPassive="0">
774   <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
775   <samlp:IDPList>
776     <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
777       Name="Identity Provider X"
778       Loc="https://IdentityProvider.example.com/saml2/sso"
779     </samlp:IDPEntry>
780     <samlp:GetComplete>
781       https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
782     </samlp:GetComplete>
783   </samlp:IDPList>
784   </ecp:Request>
785   <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
786     S:mustUnderstand="1" S:actor="http://schemas.xmlsoap.org/soap/actor/next/"
787     ...
788   </ecp:RelayState>
789 </S:Header>
790 <S:Body>
791   <samlp:AuthnRequest> ... </samlp:AuthnRequest>
792 </S:Body>
793 </S:Envelope>

```

794 4.2.4.5 ECP>IDP Request Example

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

```

798 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
799   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
800   <S:Body>
801     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
802   </S:Body>
803 </S:Envelope>

```

804 4.2.5 Steps 4,5: Authentication Response SOAP Message: IDP>ECP>SP

805 The identity provider returns a SAML <Response> message (or SOAP fault) when presented with an
806 authentication request, after having established the identity of the principal. The SAML response is
807 conveyed using the SAML SOAP binding in a SOAP message with a <Response> element in the SOAP
808 body, intended for the service provider as the ultimate SOAP receiver. The rules for the response

809 specified in the browser SSO profile in section 4.1.4.2 MUST be followed.

810 The identity provider's response message MUST contain a profile-specific ECP Response SOAP header
811 block, and MAY contain an ECP RelayState header block, both targeted at the ECP. The ECP removes
812 the header block(s), and MAY add a PAOS Response SOAP header block and an ECP RelayState
813 header block before forwarding the SOAP response to the service provider using the PAOS binding.

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

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

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

828 **4.2.5.1 ECP Response Header Block : IDP > ECP**

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

831 S:mustUnderstand [Required]

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

834 S:actor [Required]

835 The value MUST be next.

836 AssertionConsumerServiceURL [Required]

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

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

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

846 The ECP Response SOAP header has no element content.

847 See section 4.2.8 for the XML schema that defines this header block.

848 **4.2.5.2 IDP>ECP Response Example**

849 <S:Envelope

850 xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"

851 xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"

852 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

```

853 <S:Header>
854   <ecp:Response S:mustUnderstand="1" S:actor="next"
855 AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_consum
856 er"/>
857 </S:Header>
858 <S:Body>
859   <samlp:Response> ... </samlp:Response>
860 </S:Body>
861 </S:Envelope>

```

862 **4.2.5.3 PAOS Response Header Block : ECP>SP**

863 The PAOS Response header block includes the following attributes:

864 `S:mustUnderstand` [Required]

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

867 `S:actor` [Required]

868 The value **MUST** be `next`.

869 `refToMessageID` [Optional]

870 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
871 **MUST** be added by the ECP if the corresponding PAOS request specified the `messageID` attribute.

872 Note that the equivalent functionality is provided in SAML using `<AuthnRequest>` and `<Response>`
873 correlation.

874 The PAOS Response SOAP header has no element content.

875 **4.2.5.4 ECP>SP Response Example**

```

876 <S:Envelope
877   xmlns:paos="urn:liberty:paos:2003-08"
878   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
879   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
880 <S:Header>
881   <paos:Response refToMessageID="6c3a4f8b9c2d" S:actor="next" S:mustUnderstand="1"/>
882   <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
883     S:mustUnderstand="1" S:actor="next">
884     ...
885   </ecp:RelayState>
886 </S:Header>
887 <S:Body>
888   <samlp:Response> ... </samlp:Response>
889 </S:Body>
890 </S:Envelope>

```

891 **4.2.6 Step 6: HTTP service response: SP>ECP**

892 Once the service provider has received the SAML response in an HTTP request (in a SOAP envelope
893 using PAOS), it may respond with the service data in the HTTP response. In consuming the response, the

894 rules specified in the browser SSO profile in section 4.1.4.3 and 4.1.4.5 MUST be followed. That is, the
895 same processing rules used when receiving the <Response> with the HTTP POST binding apply to the
896 use of PAOS.

897 4.2.7 Security Considerations

- 898 1. The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO
899 profile, the assertions enclosed in the <Response> MUST be signed. The delivery of the response
900 in the SOAP envelope via PAOS is essentially analogous to the use of the HTTP POST binding and
901 security countermeasures appropriate to that binding are used.
- 902 2. The SOAP headers should be integrity protected, such as with SOAP Message Security or through
903 the use of SSL/TLS over every HTTP exchange with the client.
- 904 3. The service provider should be authenticated to the ECP, for example with server-side TLS
905 authentication.
- 906 4. The ECP should be authenticated to the identity provider, such as by maintaining an authenticated
907 session.

908 4.2.8 ECP Profile XML Schema

909 The following normative XML schema defines the SOAP Request/Response header blocks used by this
910 profile.

```
911 <schema
912   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
913   xmlns="http://www.w3.org/2001/XMLSchema"
914   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
915   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
916   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
917   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
918   elementFormDefault="unqualified"
919   attributeFormDefault="unqualified"
920   blockDefault="substitution"
921   version="2.0">
922   <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
923     schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
924   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
925     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
926   <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
927     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
928
929   <element name="Request" type="ecp:RequestType"/>
930   <complexType name="RequestType">
931     <sequence>
932       <element ref="saml:Issuer"/>
933       <element ref="samlp:IDPList" minOccurs="0"/>
934     </sequence>
935     <attribute ref="S:mustUnderstand" use="required"/>
936     <attribute ref="S:actor" use="required"/>
937     <attribute name="ProviderName" type="string" use="optional"/>
938     <attribute name="IsPassive" type="boolean" use="optional"/>
939   </complexType>
940
941   <element name="Response" type="ecp:ResponseType"/>
942   <complexType name="ResponseType">
943     <attribute ref="S:mustUnderstand" use="required"/>
944     <attribute ref="S:actor" use="required"/>
945     <attribute name="AssertionConsumerServiceURL" type="anyURI"
946     use="required"/>
947   </complexType>
948
949   <element name="RelayState" type="ecp:RelayStateType"/>
```

```

947     <complexType name="RelayStateType">
948         <simpleContent>
949             <extension base="string">
950                 <attribute ref="S:mustUnderstand" use="required"/>
951                 <attribute ref="S:actor" use="required"/>
952             </extension>
953         </simpleContent>
954     </complexType>
955 </schema>

```

956 **4.3 Identity Provider Discovery Profile**

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

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

966 **4.3.1 Common Domain Cookie**

967 The name of the cookie MUST be `_saml_idp`. The format of the cookie value MUST be a set of one or
968 more base-64 encoded URI values separated by a single space character. Each URI is the unique
969 identifier of an identity provider, as defined in section 8.3.6 of [SAMLCore]. The final set of values is then
970 URL encoded.

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

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

978 Cookie syntax should be in accordance with [RFC2965] or [NetscapeCookie]. The cookie MAY be either
979 session-only or persistent. This choice may be made within a deployment, but should apply uniformly to all
980 identity providers in the deployment.

981 **4.3.2 Setting the Common Domain Cookie**

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

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

992 4.3.3 Obtaining the Common Domain Cookie

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

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

999 The specific means by which the service provider reads the cookie are implementation-specific so long as
1000 it is able to cause the user agent to present cookies that have been set with the parameters given in
1001 section Section 4.3.1. One possible implementation strategy is described as follows and should be
1002 considered non-normative. Additionally, it may be sub-optimal for some applications.

- 1003 • Have previously established a DNS and IP alias for itself in the common domain.
- 1004 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1005 scheme. The structure of the URL is private to the implementation and may include session
1006 information needed to identify the user-agent.
- 1007 • Set the cookie on the redirected user agent using the parameters specified above.
- 1008 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1009 4.4 Single Logout Profile

1010 In the scenario supported by the Single Logout profile, a user has an authenticated session at one or more
1011 service providers (the session participants). The identity provider that supplied assertions to the service
1012 providers acts as (or on behalf of) the session authority. The user then wishes to terminate his or her
1013 sessions, or has their sessions administratively terminated (due to timeout, etc.). To implement this
1014 scenario, a profile of the SAML Single Logout protocol is used.

1015 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1016 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1017 front-channel binding may be required, for example, in cases in which a principal's session state exists
1018 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1019 session participant is required.

1020 4.4.1 Required Information

1021 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1022 **Contact information:** security-services-comment@lists.oasis-open.org

1023 **Description:** Given below.

1024 **Updates:** None

1025 4.4.2 Profile Overview

1026 The following figure illustrates the basic template for achieving single logout:

1027 <need figure>

1028 The following steps are described by the profile. Within an individual step, there may be one or more
1029 actual message exchanges depending on the binding used for that step and other implementation-
1030 dependent behavior.

1031 1. <LogoutRequest> issued by Service Provider to Identity Provider

1032 In step 1, the service provider initiates single logout and terminates a principal's session(s) by
1033 sending a `<LogoutRequest>` message to the identity provider from whom it received the
1034 corresponding authentication assertion. The request may be sent directly to the identity provider
1035 or sent indirectly through the user agent.

1036 **2. Identity Provider determines Session Participants**

1037 In step 2, the identity provider uses the contents of the `<LogoutRequest>` message (or if
1038 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
1039 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
1040 are repeated for each session participant identified.

1041 **3. `<LogoutRequest>` issued by Identity Provider to Session Participant/Authority**

1042 In step 3, the identity provider issues a `<LogoutRequest>` message to a session participant or
1043 session authority related to one or more of the session(s) being terminated. The request may be
1044 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
1045 request in step 1).

1046 **4. Session Participant/Authority issues `<LogoutResponse>` to Identity Provider**

1047 In step 4, a session participant or session authority terminates the principal's session(s) as
1048 directed by the request (if possible) and returns a `<LogoutResponse>` to the identity provider.
1049 The response may be returned directly to the identity provider or indirectly through the user agent
1050 (if consistent with the form of the request in step 3).

1051 **5. Identity Provider issues `<LogoutResponse>` to Service Provider**

1052 In step 5, the identity provider issues a `<LogoutResponse>` message to the original requesting
1053 service provider. The response may be returned directly to the service provider or indirectly
1054 through the user agent (if consistent with the form of the request in step 1).

1055 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1056 `<LogoutRequest>` to all session participants, also skipping step 5.

1057 **4.4.3 Profile Description**

1058 If the profile is initiated by a service provider, start with section 4.4.3.1. If initiated by the identity provider,
1059 start with section 4.4.3.2. In the descriptions below, the following is referred to:

1060 **Single Logout Service**

1061 This is the single logout protocol endpoint at an identity or service provider to which the
1062 `<LogoutRequest>` or `<LogoutResponse>` messages (or an artifact representing them) are
1063 delivered. The same or different endpoints MAY be used for requests and responses.

1064 **4.4.3.1 `<LogoutRequest>` issued by Service Provider to Identity Provider**

1065 If the logout profile is initiated by a service provider, it examines the authentication assertion(s) it received
1066 pertaining to the local session(s) being terminated, and collects the `SessionIndex` value(s) it received
1067 from the identity provider. If multiple identity providers are involved, then the profile MUST be repeated
1068 independently for each one.

1069 To initiate the profile, the service provider issues a `<LogoutRequest>` message to the identity provider's
1070 single logout service request endpoint containing one or more applicable `<SessionIndex>` elements. At
1071 least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to determine the
1072 location of this endpoint and the bindings supported by the identity provider.

1073 **Synchronous Bindings (Back-Channel)**

1074 The service provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to

1075 send the request directly to the identity provider. The identity provider would then propagate any
1076 required logout messages to additional service providers as required using a synchronous
1077 binding. The requester MUST authenticate itself to the identity provider, either by signing the
1078 <LogoutRequest> or using any other binding-supported mechanism.

1079 **Asynchronous Bindings (Front-Channel)**

1080 Alternatively, the service provider MAY (if the principal's user agent is present) use an
1081 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to
1082 send the request to the identity provider through the user agent.

1083 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is
1084 delivered to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact
1085 Resolution profile defined in section 5 is used by the identity provider, which makes a callback to
1086 the service provider to retrieve the <LogoutRequest> message, using for example the SOAP
1087 binding.

1088 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1089 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1090 <LogoutRequest> message MUST be signed if the HTTP POST or Redirect binding is used.
1091 The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1092 request issuer when the artifact is dereferenced.

1093 Each of these bindings provide a RelayState mechanism that the service provider MAY use to
1094 associate the profile exchange with the original request. The service provider SHOULD reveal as
1095 little information as possible in the RelayState value unless the use of the profile does not require
1096 such privacy measures.

1097 Profile-specific rules for the contents of the <LogoutRequest> message are included in section 4.4.4.1.

1098 **4.4.3.2 Identity Provider determines Session Participants**

1099 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1100 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1101 principal identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1102 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1103 terminated, other than the original requesting service provider (if any), as described in section 3.7.3.2 of
1104 [SAMLCore].

1105 **4.4.3.3 <LogoutRequest> issued by Identity Provider to Session 1106 Participant/Authority**

1107 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1108 participant in a session being terminated. The request is sent in the same fashion as described in step 1
1109 using a SAML binding consistent with the capability of the responder and the availability of the user agent
1110 at the identity provider.

1111 Profile-specific rules for the contents of the <LogoutRequest> message are included in section 4.4.4.1.

1112 **4.4.3.4 Session Participant/Authority issues <LogoutResponse> to Identity 1113 Provider**

1114 The session participant/authority MUST process the <LogoutRequest> message as defined in
1115 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1116 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1117 to complete the SAML protocol exchange.

1118 **Synchronous Bindings (Back-Channel)**

1119 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1120 response is returned directly to complete the synchronous communication. The responder MUST
1121 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1122 using any other binding-supported mechanism.

1123 **Asynchronous Bindings (Front-Channel)**

1124 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1125 Artifact bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the
1126 user agent to the identity provider's single logout service response endpoint. Metadata (as in
1127 [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings supported
1128 by the identity provider.

1129 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is
1130 delivered to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact
1131 Resolution profile defined in section 5 is used by the identity provider, which makes a callback to
1132 the responding entity to retrieve the <LogoutResponse> message, using for example the SOAP
1133 binding.

1134 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1135 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1136 <LogoutResponse> message MUST be signed if the HTTP POST or Redirect binding is used.
1137 The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1138 response issuer when the artifact is dereferenced.

1139 Profile-specific rules for the contents of the <LogoutResponse> message are included in section
1140 4.4.4.2.

1141 **4.4.3.5 Identity Provider issues <LogoutResponse> to Service Provider**

1142 After processing the original service provider's <LogoutRequest> in step 1, or upon encountering an
1143 error, the identity provider MUST respond to the original request with a <LogoutResponse> containing
1144 an appropriate status code to complete the SAML protocol exchange.

1145 The response is sent to the original service provider in the same fashion as described in step 4, using a
1146 SAML binding consistent with the binding used in the request, the capability of the responder, and the
1147 availability of the user agent at the identity provider.

1148 Profile-specific rules for the contents of the <LogoutResponse> message are included in section
1149 4.4.4.2.

1150 **4.4.4 Use of Single Logout Protocol**

1151 **4.4.4.1 <LogoutRequest> Usage**

1152 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1153 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1154 format:entity.

1155 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1156 the message or using a binding-specific mechanism.

1157 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1158 the authentication assertion the requester issued or received regarding the session being terminated, per
1159 the matching rules defined in section 3.3.4 of [SAMLCore].

1160 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the
1161 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such
1162 elements to indicate the termination of all of the principal's applicable sessions.

1163 4.4.4.2 <LogoutResponse> Usage

1164 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1165 entity; the Format attribute MUST be omitted or have a value of
1166 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1167 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1168 the message or using a binding-specific mechanism.

1169 4.4.5 Use of Metadata

1170 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported
1171 bindings and location(s) to which an entity may send requests and responses using this profile.

1172 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1173 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1174 settings to use, along with a public key to use in delivering a bulk encryption key.

1175 4.5 Name Identifier Management Profile

1176 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1177 some form of persistent identifier for a principal with a service provider, allowing them to share a common
1178 identifier for some length of time. Subsequently, the identity provider may wish to notify the service
1179 provider of a change in the format and/or value that it will use to identify the same principal in the future.
1180 Alternatively the service provider may wish to attach its own "alias" for the principal in order to insure that
1181 the identity provider will include it when communicating with it in the future about the principal. Finally, one
1182 of the providers may wish to inform the other that it will no longer issue or accept messages using a
1183 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management
1184 protocol is used.

1185 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1186 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1187 front-channel binding may be required, for example, in cases in which direct interaction between the user
1188 agent and the responding provider is required in order to effect the change.

1189 4.5.1 Required Information

1190 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1191 **Contact information:** security-services-comment@lists.oasis-open.org

1192 **Description:** Given below.

1193 **Updates:** None

1194 4.5.2 Profile Overview

1195 The following figure illustrates the basic template for the name identifier management profile.

1196 <need figure>

1197 The following steps are described by the profile. Within an individual step, there may be one or more
1198 actual message exchanges depending on the binding used for that step and other implementation-
1199 dependent behavior.

1200 1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider

1201 In step 1, an identity or service provider initiates the profile by sending a

1202 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
1203 The request may be sent directly to the responding provider or sent indirectly through the user
1204 agent.

1205 **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1206 In step 2, the responding provider (after processing the request) issues a
1207 <ManageNameIDResponse> message to the original requesting provider. The response may be
1208 returned directly to the requesting provider or indirectly through the user agent (if consistent with
1209 the form of the request in step 1).

1210 **4.5.3 Profile Description**

1211 In the descriptions below, the following is referred to:

1212 **Name Identifier Management Service**

1213 This is the name identifier management protocol endpoint at an identity or service provider to
1214 which the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
1215 representing them) are delivered. The same or different endpoints MAY be used for requests and
1216 responses.

1217 **4.5.3.1 <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1218 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
1219 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1220 used to determine the location of this endpoint and the bindings supported by the responding provider.

1221 **Synchronous Bindings (Back-Channel)**

1222 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind],
1223 to send the request directly to the other provider. The requester MUST authenticate itself to the
1224 other provider, either by signing the <ManageNameIDRequest> or using any other binding-
1225 supported mechanism.

1226 **Asynchronous Bindings (Front-Channel)**

1227 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1228 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to
1229 send the request to the other provider through the user agent.

1230 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1231 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact
1232 Resolution profile defined in section 5 is used by the other provider, which makes a callback to the
1233 requesting provider to retrieve the <ManageNameIDRequest> message, using for example the
1234 SOAP binding.

1235 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1236 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1237 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1238 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating
1239 the request issuer when the artifact is dereferenced.

1240 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1241 associate the profile exchange with the original request. The requesting provider SHOULD reveal
1242 as little information as possible in the RelayState value unless the use of the profile does not
1243 require such privacy measures.

1244 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in section
1245 4.4.4.1.

1246 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service**
1247 **Provider**

1248 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1249 processing the message or upon encountering an error, the recipient MUST issue a
1250 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1251 to complete the SAML protocol exchange.

1252 **Synchronous Bindings (Back-Channel)**

1253 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind],
1254 the response is returned directly to complete the synchronous communication. The responder
1255 MUST authenticate itself to the requesting provider, either by signing the
1256 <ManageNameIDResponse> or using any other binding-supported mechanism.

1257 **Asynchronous Bindings (Front-Channel)**

1258 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1259 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned
1260 through the user agent to the requesting provider's name identifier management service response
1261 endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint
1262 and the bindings supported by the requesting provider.

1263 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1264 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1265 Resolution profile defined in section 5 is used by the requesting provider, which makes a callback
1266 to the responding provider to retrieve the <ManageNameIDResponse> message, using for
1267 example the SOAP binding.

1268 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1269 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1270 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is
1271 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating
1272 the response issuer when the artifact is dereferenced.

1273 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1274 section 4.4.4.2.

1275 **4.5.4 Use of Name Identifier Management Protocol**

1276 **4.5.4.1 <ManageNameIDRequest> Usage**

1277 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1278 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1279 format:entity.

1280 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1281 the message or using a binding-specific mechanism.

1282 **4.5.4.2 <ManageNameIDResponse> Usage**

1283 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1284 entity; the Format attribute MUST be omitted or have a value of
1285 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1286 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1287 the message or using a binding-specific mechanism.

1288 **4.5.5 Use of Metadata**

1289 [SAMLMeta] defines an endpoint element, `<md:ManageNameIDService>`, to describe supported
1290 bindings and location(s) to which an entity may send requests and responses using this profile.

1291 A requester, if encrypting the principal's identifier, can use the responder's `<md:KeyDescriptor>`
1292 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1293 settings to use, along with a public key to use in delivering a bulk encryption key.

1294 5 Artifact Resolution Profile

1295 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1296 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1297 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1298 binding, such as the SOAP binding defined in [SAMLBind].

1299 5.1 Required Information

1300 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1301 **Contact information:** security-services-comment@lists.oasis-open.org

1302 **Description:** Given below.

1303 **Updates:** None

1304 5.2 Profile Overview

1305 The message exchange and basic processing rules that govern this profile are largely defined by section
1306 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1307 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1308 SOAP 1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1309 The following figure illustrates the basic template for the artifact resolution profile.

1310 <need figure>

1311 The following steps are described by the profile.

1312 1. <ArtifactResolve> issued by Requesting Entity

1313 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1314 artifact issuer.

1315 2. <ArtifactResponse> issued by Responding Entity

1316 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1317 message to the requester.

1318 5.3 Profile Description

1319 In the descriptions below, the following is referred to:

1320 **Artifact Resolution Service**

1321 This is the artifact resolution protocol endpoint at an artifact issuer to which
1322 <ArtifactResolve> messages are delivered.

1323 5.3.1 <ArtifactResolve> issued by Requesting Entity

1324 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1325 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact
1326 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1327 endpoint and the bindings supported by the artifact issuer

1328 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1329 request directly to the artifact issuer. The requester SHOULD authenticate itself to the identity provider,
1330 either by signing the <ArtifactResolve> message or using any other binding-supported mechanism.
1331 Specific profiles that use the HTTP Artifact binding MAY impose additional requirements such that
1332 authentication is mandatory.

1333 Profile-specific rules for the contents of the <ArtifactResolve> message are included in section 5.4.1.

1334 **5.3.2 <ArtifactResponse> issued by Responding Entity**

1335 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After
1336 processing the message or upon encountering an error, the artifact issuer MUST return an
1337 <ArtifactResponse> message containing an appropriate status code to the requester to complete the
1338 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1339 artifact will also be included.

1340 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or
1341 using any other binding-supported mechanism.

1342 Profile-specific rules for the contents of the <ArtifactResponse> message are included in section
1343 5.4.2.

1344 **5.4 Use of Artifact Resolution Protocol**

1345 **5.4.1 <ArtifactResolve> Usage**

1346 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1347 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1348 format:entity.

1349 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1350 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1351 binding MAY impose additional requirements such that authentication is mandatory.

1352 **5.4.2 <ArtifactResponse> Usage**

1353 The <Issuer> element MUST be present and MUST contain the unique identifier of the artifact issuer;
1354 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1355 format:entity.

1356 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1357 the message or using a binding-specific mechanism.

1358 **5.5 Use of Metadata**

1359 [SAMLMeta] defines an indexed endpoint element, <md:ArtifactResolutionService>, to describe
1360 supported bindings and location(s) to which a requester may send requests using this profile. The index
1361 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1362 EndpointIndex field.

1363 6 Assertion Query/Request Profile

1364 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1365 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1366 synchronous binding, such as the SOAP binding defined in [SAMLBind].

1367 6.1 Required Information

1368 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1369 **Contact information:** security-services-comment@lists.oasis-open.org

1370 **Description:** Given below.

1371 **Updates:** None.

1372 6.2 Profile Overview

1373 The message exchange and basic processing rules that govern this profile are largely defined by section
1374 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1375 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1376 SOAP 1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1377 The following figure illustrates the basic template for the query/request profile.

1378 <need figure>

1379 The following steps are described by the profile.

1380 1. Query/Request issued by Requesting Entity

1381 In step 1, a requester initiates the profile by sending an <AssertionIDRequest>,
1382 <SubjectQuery>, <AuthnQuery>, <AttributeQuery>, Or <AuthzDecisionQuery>
1383 message to a SAML authority.

1384 2. <Response> issued by SAML Authority

1385 In step 2, the responding SAML authority (after processing the query or request) issues a
1386 <Response> message to the requester.

1387 6.3 Profile Description

1388 In the descriptions below, the following are referred to:

1389 Query/Request Service

1390 This is the query/request protocol endpoint at a SAML authority to which query or
1391 <AssertionIDRequest> messages are delivered.

1392 6.3.1 Query/Request issued by Requesting Entity

1393 To initiate the profile, a requester issues an <AssertionIDRequest>, <SubjectQuery>,
1394 <AuthnQuery>, <AttributeQuery>, Or <AuthzDecisionQuery> message to a SAML authority's
1395 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1396 this endpoint and the bindings supported by the SAML authority.

1397 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1398 request directly to the identity provider. The requester SHOULD authenticate itself to the SAML authority
1399 either by signing the message or using any other binding-supported mechanism.

1400 Profile-specific rules for the contents of the various messages are included in section 6.4.1.

1401 **6.3.2 <Response> issued by SAML Authority**

1402 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1403 processing the message or upon encountering an error, the SAML authority MUST return a <Response>
1404 message containing an appropriate status code to the requester to complete the SAML protocol
1405 exchange. If the request is successful in locating one or more matching assertions, they will also be
1406 included in the response.

1407 The responder SHOULD authenticate itself to the requester, either by signing the <Response> or using
1408 any other binding-supported mechanism.

1409 Profile-specific rules for the contents of the <Response> message are included in section 6.4.2.

1410 **6.4 Use of Query/Request Protocol**

1411 **6.4.1 Query/Request Usage**

1412 The <Issuer> element MUST be present.

1413 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1414 signing the message or using a binding-specific mechanism.

1415 **6.4.2 <Response> Usage**

1416 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1417 SAML authority; the Format attribute MUST be omitted or have a value of
1418 urn:oasis:names:tc:SAML:2.0:nameid-format:entity. Note that this need not necessarily
1419 match the <Issuer> element in the returned assertion(s).

1420 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1421 signing the message or using a binding-specific mechanism.

1422 **6.5 Use of Metadata**

1423 [SAMLMeta] defines several endpoint elements, <md:AssertionIDRequestService>,
1424 <md:AuthnQueryService>, <md:AttributeService>, and <md:AuthzService>, to describe
1425 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1426 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1427 use that entity's <md:KeyDescriptor> element with a use attribute of encryption to determine an
1428 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1429 encryption key.

1430 7 Name Identifier Mapping Profile

1431 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1432 different name identifier for the same principal. This profile describes the use of this protocol with a
1433 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1434 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

1435 7.1 Required Information

1436 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1437 **Contact information:** security-services-comment@lists.oasis-open.org

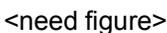
1438 **Description:** Given below.

1439 **Updates:** None.

1440 7.2 Profile Overview

1441 The message exchange and basic processing rules that govern this profile are largely defined by section
1442 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1443 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1444 SOAP 1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1445 The following figure illustrates the basic template for the name identifier mapping profile.

1446  <need figure>

1447 The following steps are described by the profile.

1448 1. <NameIDMappingRequest> issued by Requesting Entity

1449 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1450 an identity provider.

1451 2. <NameIDMappingResponse> issued by Identity Provider

1452 In step 2, the responding identity provider (after processing the request) issues a
1453 <NameIDMappingResponse> message to the requester.

1454 7.3 Profile Description

1455 In the descriptions below, the following is referred to:

1456 Name Identifier Mapping Service

1457 This is the name identifier mapping protocol endpoint at an identity provider to which
1458 <NameIDMappingRequest> messages are delivered.

1459 7.3.1 <NameIDMappingRequest> issued by Requesting Entity

1460 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1461 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1462 location of this endpoint and the bindings supported by the identity provider.

1463 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the

1464 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1465 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1466 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1467 section 7.4.1.

1468 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1469 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1470 After processing the message or upon encountering an error, the identity provider MUST return a
1471 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1472 complete the SAML protocol exchange.

1473 The responder MUST authenticate itself to the requester, either by signing the
1474 <NameIDMappingResponse> or using any other binding-supported mechanism.

1475 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1476 section 7.4.2.

1477 **7.4 Use of Name Identifier Mapping Protocol**

1478 **7.4.1 <NameIDMappingRequest> Usage**

1479 The <Issuer> element MUST be present.

1480 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1481 the message or using a binding-specific mechanism.

1482 **7.4.2 <NameIDMappingResponse> Usage**

1483 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1484 identity provider; the Format attribute MUST be omitted or have a value of
1485 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1486 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1487 the message or using a binding-specific mechanism.

1488 Section 2.3.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1489 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1490 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it
1491 in subsequent protocol messages or assertions.

1492 **7.4.2.1 Limiting Use of Mapped Identifier**

1493 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1494 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1495 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1496 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1497 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1498 parties, and MUST be signed for integrity protection.

1499 **7.5 Use of Metadata**

1500 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1501 bindings and location(s) to which a requester may send requests using this profile.

1502 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1503 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1504 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1505 8 SAML Attribute Profiles

1506 8.1 Basic Attribute Profile

1507 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with
1508 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas
1509 to validate syntax.

1510 8.1.1 Required Information

1511 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1512 **Contact information:** security-services-comment@lists.oasis-open.org

1513 **Description:** Given below.

1514 **Updates:** None.

1515 8.1.2 SAML Attribute Naming

1516 The `NameFormat` XML attribute in `<AttributeDesignator>` and `<Attribute>` elements MUST be
1517 `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`.

1518 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1519 8.1.3 Profile-Specific XML Attributes

1520 No additional XML attributes are defined for use with the `<AttributeDesignator>` or `<Attribute>`
1521 elements.

1522 8.1.4 `<AttributeDesignator>` Comparison

1523 Two `<AttributeDesignator>` elements are equal if and only if the values of their `Name` XML attributes
1524 are equal in the sense of Section 3.3.6 of [XML-Schema-Part2].

1525 8.1.5 SAML Attribute Values

1526 The schema type of the contents of the `<AttributeValue>` element MUST be drawn from one of the
1527 types defined in Section 3.3 of [XML-Schema-Part2]. The `xsi:type` attribute MUST be present and be
1528 given the appropriate value.

1529 8.1.6 Example

1530 TBD

1531 8.2 X.500/LDAP Attribute Profile

1532 There is a substantial body of work describing standard syntaxes for X.500/LDAP attributes. This includes
1533 RFC2256 [RFC2256], which describes an overview of the attribute types and object classes defined by the
1534 ISO and ITU-T committees in the X.500 documents, in particular those intended for use by directory
1535 clients. Several authors have built upon these approaches to develop additional attribute types and some
1536 of these have been widely implemented. For example, the `inetOrgPerson` object class defined in
1537 RFC2798 [RFC2798] has received wide implementation amongst LDAP vendors. Other efforts include the

1538 definition of the `eduPerson` object class by the EDUCAUSE/Internet2 task force [`eduPersonSchema`].
1539 The X.500/LDAP attribute profile standardizes the naming and representation of such attributes when
1540 expressed as SAML attributes, providing unique naming consistent with the OID mechanism used natively
1541 by such specifications.

1542 **8.2.1 Required Information**

1543 **Identification:** `urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP`

1544 **Contact information:** security-services-comment@lists.oasis-open.org

1545 **Description:** Given below.

1546 **Updates:** None.

1547 **8.2.2 SAML Attribute Naming**

1548 The `NameFormat` XML attribute in `<AttributeDesignator>` and `<Attribute>` elements MUST be
1549 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1550 To construct attribute names, the URN `oid` namespace described in [RFC3061] is used. In this approach
1551 the `Name` XML attribute is based on the OID assigned to the X.500/LDAP attribute type.

1552 **Example:**

1553 `urn:oid:1.3.6.1.4.1.299`

1554 X.500 conventions require that every object-class be identified with a unique OID. This ensures that
1555 attribute names are unambiguous.

1556 For purposes of human readability, there may also be a requirement for some applications to carry an
1557 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1558 [SAMLCore]) MAY be used for this purpose.

1559 **8.2.3 Profile-Specific XML Attributes**

1560 No additional XML attributes are defined for use with the `<AttributeDesignator>` or `<Attribute>`
1561 elements.

1562 **8.2.4 <AttributeDesignator> Comparison**

1563 Two `<AttributeDesignator>` elements are equal if and only if their `Name` XML attribute values are
1564 equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the comparison.

1565 **8.2.5 SAML Attribute Values**

1566 The canonical representation for X.500/LDAP attribute syntaxes is an octet string. However, to simplify the
1567 job of the attribute consumer, any X.500/LDAP attribute syntax that can easily be expressed in string form
1568 SHOULD be passed as a string within the `<AttributeValue>` element, with no additional whitespace.
1569 In such cases, the `xsi:type` XML attribute MUST be set to `xsd:string`.

1570 Examples of such attribute syntaxes are those with string, numeric, or date/time values. Date/time values
1571 MUST be expressed in UTC form.

1572 Any attribute value (particularly those without a reasonable string form, such as binary data) MAY be
1573 passed by base64-encoding the octet string and specifying an `xsi:type` XML attribute of
1574 `xsd:base64Binary`. The `xsi:type` XML attribute MUST be present in such cases.

1575 As additional standards in the expression of X.500/LDAP attribute syntaxes in XML form develop, this
1576 profile may evolve to incorporate such approaches.

1577 8.2.6 Example

1578 TBD

1579 8.3 UUID Attribute Profile

1580 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1581 values. It is applicable when the attribute's source system is one that identifies an attribute with a UUID.
1582 The value of the attribute may also be a UUID, but need not be.

1583 8.3.1 Required Information

1584 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1585 **Contact information:** security-services-comment@lists.oasis-open.org

1586 **Description:** Given below.

1587 **Updates:** None.

1588 8.3.2 UUID and GUID Background

1589 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1590 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs
1591 were originally used in the Network Computing System (NCS), and then used in the Open Software
1592 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1593 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1594 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1595 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1596 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1597 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1598 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1599 "friendly name". For instance the above UUID could represent the user john.hughes@entegrity.com.

1600 8.3.3 SAML Attribute Naming

1601 The NameFormat XML attribute in <AttributeDesignator> and <Attribute> elements MUST be
1602 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1603 To construct attribute names, the URN `uuid` namespace described in [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt>] is used. In this approach the Name XML attribute is based on the
1604 URN form of the underlying UUID that identifies the attribute.
1605

1606 Example:

1607 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1608 For purposes of human readability, there may also be a requirement for some applications to carry an
1609 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1610 [SAMLCore]) MAY be used for this purpose.

1611 8.3.4 Profile-Specific XML Attributes

1612 No additional XML attributes are defined for use with the <AttributeDesignator> or <Attribute>
1613 elements.

1614 **8.3.5 <AttributeDesignator> Comparison**

1615 Two <AttributeDesignator> elements are equal if and only if their Name XML attribute values are
1616 equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1617 FriendlyName attribute plays no role in the comparison.

1618 **8.3.6 SAML Attribute Values**

1619 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1620 used to express the value within the <AttributeValue> element. The xsi:type XML attribute MUST
1621 be set to xsd:anyURI.

1622 If the attribute's value is not a UUID, then there are no restrictions on the use of the <AttributeValue>
1623 element.

1624 **8.3.7 Standard DCE Attributes**

1625 DCE is able to transport a wide range of authorization data within its Privilege Attribute Certificate (PAC).
1626 Several useful attributes carried within the PAC structure receive special mention. Each is named by a
1627 well-defined UUID value, as described below:

1628 **8.3.7.1 Principal**

1629 This single-valued attribute represents the SAML subject's DCE principal identity, in UUID form.

1630 **Name:** urn:uuid:TBD

1631 **<AttributeValue>:** a UUID URN containing the UUID of the DCE principal

1632 **8.3.7.2 Primary Group**

1633 This single-valued attribute represents the SAML subject's primary DCE group membership, in UUID
1634 form.

1635 **Name:** urn:uuid:TBD

1636 **<AttributeValue>:** a UUID URN containing the UUID of the DCE principal's primary DCE group

1637 **8.3.7.3 Groups**

1638 This multi-valued attribute represents the SAML subject's DCE local group memberships, in UUID form.

1639 **Name:** urn:uuid:TBD

1640 **<AttributeValue>:** a UUID URN containing the UUID of a group in which the DCE principal is a
1641 member

1642 **8.3.8 Example**

1643 TBD

1644 **8.4 XACML Attribute Profile**

1645 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
1646 eXtensible Access Control Markup Language (XACML) standard specification [XACML]. Since the SAML
1647 attribute format differs from the XACML attribute format, there is a mapping that must be performed. The

1648 XACML attribute profile facilitates this mapping by standardizing naming, value syntax, and additional
1649 attribute metadata. SAML attributes generated in conformance with this profile can be mapped
1650 automatically into XACML attributes and used as input to XACML authorization decisions.

1651 8.4.1 Required Information

1652 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML

1653 **Contact information:** security-services-comment@lists.oasis-open.org

1654 **Description:** Given below.

1655 **Updates:** None.

1656 8.4.2 SAML Attribute Naming

1657 The `NameFormat` XML attribute in `<AttributeDesignator>` and `<Attribute>` elements MUST be
1658 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1659 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1660 For purposes of human readability, there may also be a requirement for some applications to carry an
1661 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1662 [SAMLCore]) MAY be used for this purpose, but is not translatable into the XACML attribute equivalent.

1663 8.4.3 Profile-Specific XML Attributes

1664 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
1665 valued XML attribute called `DataType` is defined in the XML namespace
1666 urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML.

1667 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified
1668 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

1669 While in principle any URI reference can be used as a data type, the standard values to be used are
1670 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
1671 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
1672 must be extended to support the new data types.

1673 8.4.4 <AttributeDesignator> Comparison

1674 Two `<AttributeDesignator>` elements are equal if and only if their `Name` XML attribute values are
1675 equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

1676 8.4.5 SAML Attribute Values

1677 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed
1678 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
1679 types corresponding to the types defined in section 3.3 of [XML-Schema-Part2], the `xsi:type` XML
1680 attribute SHOULD also be used.

1681 8.4.6 Profile-Specific Schema

1682 The following schema defines the profile-specific `DataType` XML attribute:

```
1683 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
1684         xmlns="http://www.w3.org/2001/XMLSchema"  
1685         version="2.0">  
1686     <attribute name="DataType" type="anyURI"/>
```

1687 </schema>

1688 **8.4.7 Example**

1689 TBD

9 References

1690

- 1691 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 1692 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,
1693 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 1694 **[CoreAssnEx]** Core Assertions Architecture, Examples and Explanations, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf)
1695 [open.org/committees/security/docs/draft-sstc-core-phill-07.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf).
- 1696 **[HTML401]** HTML 4.01 Specification, W3C Recommendation 24 December 1999,
1697 <http://www.w3.org/TR/html4>.
- 1698 **[Liberty]** The Liberty Alliance Project, <http://www.projectliberty.org>.
- 1699 **[MSURL]** Microsoft technical support article,
1700 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 1701 **[PAOS]** Aarts, R., “Liberty Reverse HTTP Binding for SOAP Specification”, Version: 1.0,
1702 <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>
- 1703 **[Rescorla-Sec]** E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations,
1704 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 1705 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 1706 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 1707 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 1708 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
1709 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 1710 **[RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119,
1711 March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 1712 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 1713 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 1714 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 1715 **[RFC2617]** HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617,
1716 <http://www.ietf.org/rfc/rfc2617.txt>.
- 1717 **[SAMLBind]** Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, DRAFT
- 1718 **[SAMLCore]** E. Maler et al. Assertions and Protocol for the OASIS Security Assertion Markup
1719 Language (SAML). OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1.
1720 <http://www.oasis-open.org/committees/security/>.
- 1721 **[SAMLMeta]** Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0, DRAFT
- 1722 **[SAMLGloss]** E. Maler et al. Glossary for the OASIS Security Assertion Markup Language (SAML).
1723 OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1. [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1724 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1725 **[SAMLSec]** E. Maler et al. Security Considerations for the OASIS Security Assertion Markup
1726 Language (SAML), OASIS, September 2003, Document ID oasis-sstc-saml-sec-consider-
1727 1.1. <http://www.oasis-open.org/committees/security/>.
- 1728 **[SAMLReqs]** Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002,
1729 <http://www.oasis-open.org/committees/security/>.
- 1730 **[SAMLWeb]** OASIS Security Services Technical Committee website, [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1731 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1732 **[SESSION]** RL “Bob” Morgan, Support of target web server sessions in Shibboleth,
1733 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>

- 1734 **[ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1,
1735 <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- 1736 **[SOAP1.1]** D. Box et al., Simple Object Access Protocol (SOAP) 1.1, World Wide Web Consortium
1737 Note, May 2000, <http://www.w3.org/TR/SOAP>.
- 1738 **[SSL3]** A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November 1996.
- 1739 **[SSTC-Lib1.1]** Liberty v1.1 specification set submittal letter", 11 Apr 2003, <http://lists.oasis-open.org/archives/security-services/200304/msg00072.html>
- 1741 **[SSTC-Lib1.2]** Liberty ID-FF 1.2 Contribution to SSTC", 13 Nov 2003, <http://www.oasis-open.org/archives/security-services/200311/msg00060.html>
- 1743 **[WEBSO]** RL "Bob" Morgan, Interactions between Shibboleth and local-site web sign-on services,
1744 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt>
- 1745 **[WSDL1_1]** "Web Services Description Language (WSDL) 1.1", W3C Note 15 March 2001,
1746 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 1747 **[WSS]** A Nadalin, C Kaler, P Hallam-Baker, R Monzillo, "Web Services Security: SOAP Message
1748 Security .0 (WS-Security 2004)", OASIS, March 2004, <http://www.oasis-open.org/committees/wss>.
- 1750 **[WSS-SAML]** P. Hallam-Baker et al., Web Services Security: SAML Token Profile, OASIS, March 2003,
1751 <http://www.oasis-open.org/committees/wss>.
- 1752 **[XMLSig]** D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web Consortium,
1753 <http://www.w3.org/TR/xmlsig-core/>.
- 1754 **[RFC2256]** M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3,
1755 December 1997
- 1756 **[Morgan]** R. L. Morgan, Conventions for Use of X.500/LDAP Attribute Types in SAML, Draft 00, 5
1757 November 2003, available from SAML Document repository as draft-morgan-saml-attrib-
1758 x500-00.pdf
- 1759 **[RFC2798]** W. Smith, Definition of the inetOrgPerson LDAP Object class, April 2000
- 1760 **[eduPersonSchema]** eduPerson.Idif, Available from <http://www.educase.edu/eduperson>
- 1761 **[Mealling]** P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03.
1762 January 2004
- 1763 **[RFC3061]** M. Mealling, RFC3061 – A URN Namespace of Object Identifiers, Feb 2001
- 1764 **[XML-Schema-Part2]** Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes, W3C
1765 Recommendation 02 May 2001, <http://www.w3.org/TR/xmlschema-2/>
- 1766 **[XACML]** T. Moses, ed., OASIS *eXtensible Access Control Markup Language (XACML) Versions*
1767 *1.0, 1.1, and 2.0*. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
1768

1769 **A. Acknowledgments**

1770 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1771 Committee, whose voting members at the time of publication were:

- 1772 • TBD

B. Revision History

Rev	Date	By Whom	What
1	16 Feb 2004	Frederick Hirsch	Split new profiles document from bindings and profiles, removed bindings section. Added ECP profile, added and formatted references.
2	2 Mar 2004	Frederick Hirsch	Removed URL Size restriction section – this is located in the bindings document. Minor cleanup in section 2.1
3	27 Mar 2004	Frederick Hirsch	Changes to reflect core 8, review comments, corrections.
4	30 Mar 2004	Frederick Hirsch	Additional review comments, corrections.
6	16 Apr 2004	Scott Cantor	Replaced 1.1 SSO profiles with new proposal, added discovery profile, revised confirmation method descriptions, removed binding-related duplications, added placeholders for additional profiles.
7	9 May 2004	Scott Cantor	Added NameIdentifierMapping profile
8	14 May 2004	Frederick Hirsch	Changes based on 5/11/04 SSTC conference call – replace Identifier with ID in elements, in elements and attributes replace Authentication with Authn . Specifically, changed <AuthenticationStatement>, <NameIdentifierMappingRequest>, <NameIdentifierMappingResponse>, <EncryptedIdentifier>, <NameIdentifierMappingService>
9	30 May 2004	Scott Cantor	Sync'd confirmation data sections to new schema in core-14, relaxed NameIDMapping profile requirement for SOAP binding, started clean-up of ECP, adjusted SSO profile to reflect bindings-12, added back sender-vouches.
10	7 Jun 2004	Prateek Mishra	Added attribute profiles materials from hughes-mishra-baseline-attributes-04 with John Hughes updates
11	13 Jun 2004	Scott Cantor	Added metadata considerations to profiles, minor editorial cleanups, new section headers for profiles
12	7 Jun 2004	Scott Cantor	Final SSO cleanup, formalized ECP schema, fixed examples, intro section.
13	7 Jul 2004	Scott Cantor	Added RelayState ECP header, more ECP cleanup, added SingleLogout profile, fixes to discovery profile
14	7 Jul 2004	Scott Cantor	Filled in remaining profiles, re-edited attribute profiles
15	13 Jul 2004	Eve Maler	Final cleanup in preparation for last-call working draft publication.

1774

C. Notices

1775 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1776 might be claimed to pertain to the implementation or use of the technology described in this document or
1777 the extent to which any license under such rights might or might not be available; neither does it represent
1778 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1779 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1780 available for publication and any assurances of licenses to be made available, or the result of an attempt
1781 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1782 users of this specification, can be obtained from the OASIS Executive Director.

1783 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1784 other proprietary rights which may cover technology that may be required to implement this specification.
1785 Please address the information to the OASIS Executive Director.

1786 **Copyright © OASIS Open 2003-2004. All Rights Reserved.**

1787 This document and translations of it may be copied and furnished to others, and derivative works that
1788 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1789 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1790 this paragraph are included on all such copies and derivative works. However, this document itself may
1791 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1792 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1793 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1794 into languages other than English.

1795 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1796 or assigns.

1797 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1798 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1799 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1800 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1801 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and
1802 other countries.