

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

Working Draft ~~076~~, ~~824~~ DecemberFebruarySeptember
2015690

Document identifier:

sstc-saml-profiles-errata-2.0-wd-0756

Location:

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

Editors:

John Hughes, Atos Origin
Scott Cantor, Internet2
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
Prateek Mishra, Principal Identity
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems (errata editor)
[Eric Goodman, Individual \(errata editor\)](#)

Contributors to the Errata:

Rob Philpott, EMC Corporation
Nick Ragouzis, Enosis Group
Thomas Wisniewski, Entrust
Greg Whitehead, HP
Heather Hinton, IBM
Connor P. Cahill, Intel
Scott Cantor, Internet2
Nate Klingenstein, Internet2
RL 'Bob' Morgan, Internet2
John Bradley, Individual
Jeff Hodges, Individual
Joni Brennan, Liberty Alliance
Eric Tiffany, Liberty Alliance
Thomas Hardjono, M.I.T.
Tom Scavo, NCSA
Peter Davis, NeuStar, Inc.
Frederick Hirsch, Nokia Corporation
Paul Madsen, NTT Corporation
Ari Kermaier, Oracle Corporation
Hal Lockhart, Oracle Corporation
Prateek Mishra, Oracle Corporation
Brian Campbell, Ping Identity
Anil Saldhana, Red Hat Inc.
Jim Lien, RSA Security
Jahan Moreh, Sigaba
Kent Spaulding, Skyworth TTG Holdings Limited
Emily Xu, Sun Microsystems

David Staggs, Veteran's Health Administration

SAML V2.0 Contributors:

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

Abstract:

The SAML V2.0 Profiles specification defines profiles for the use of SAML assertions and request-response messages in communications protocols and frameworks, as well as profiles for SAML attribute value syntax and naming conventions. This document, known as an “errata composite”, combines corrections to reported errata with the original specification text. By design, the corrections are limited to clarifications of ambiguous or conflicting specification text. This document shows deletions from the original specification as struck-through text, and additions as colored underlined text. The “[*Enn*” designations embedded in the text refer to particular errata and their dispositions.

Status:

This errata composite document is a **working draft** based on the [original](#) OASIS Standard document that had been produced by the Security Services Technical Committee and approved by the OASIS membership on 1 March 2005. While the errata corrections appearing here are non-normative, they reflect changes specified by the Approved Errata document (currently at Working Draft revision 02), which is on an OASIS standardization track. In case of any discrepancy between this document and the Approved Errata, the latter has precedence. [See also the Errata Working Document \(currently at revision 39\), which provides background on the changes specified here.](#)

This document includes corrections for errata E12, E14, E17, E18, E20, E22, E26, E27, E32, E35, E38, E39, E40, E47, E48, E51, E52, E53, E54, E55, E56, E58, [and E63, E70, E71, and E74, E85, E90, and E93.](#)

Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them by following the instructions at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

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

Table of Contents

107		
108	1 Introduction.....	8
109	1.1 Profile Concepts.....	8
110	1.2 Notation.....	8
111	2 Specification of Additional Profiles.....	11
112	2.1 Guidelines for Specifying Profiles.....	11
113	2.2 Guidelines for Specifying Attribute Profiles.....	11
114	3 Confirmation Method Identifiers.....	13
115	3.1 Holder of Key.....	13
116	3.2 Sender Vouches.....	14
117	3.3 Bearer.....	14
118	4 SSO Profiles of SAML.....	15
119	4.1 Web Browser SSO Profile.....	15
120	4.1.1 Required Information.....	15
121	4.1.2 Profile Overview.....	15
122	4.1.3 Profile Description.....	17
123	4.1.3.1 HTTP Request to Service Provider.....	17
124	4.1.3.2 Service Provider Determines Identity Provider.....	17
125	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	17
126	4.1.3.4 Identity Provider Identifies Principal.....	18
127	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	18
128	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	19
129	4.1.4 Use of Authentication Request Protocol.....	19
130	4.1.4.1 <AuthnRequest> Usage.....	19
131	4.1.4.2 <Response> Usage.....	19
132	4.1.4.3 <Response> Message Processing Rules.....	21
133	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	21
134	4.1.4.5 POST-Specific Processing Rules.....	22
135	4.1.5 Unsolicited Responses.....	22
136	4.1.6 [E90] Use of Relay State.....	22
137	4.1.7 Use of Metadata.....	22
138	4.2 Enhanced Client or Proxy (ECP) Profile.....	23
139	4.2.1 Required Information.....	23
140	4.2.2 Profile Overview.....	24
141	4.2.3 Profile Description.....	26
142	4.2.3.1 ECP issues HTTP Request to Service Provider.....	26
143	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	27
144	4.2.3.3 ECP Determines Identity Provider.....	27
145	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	27
146	4.2.3.5 Identity Provider Identifies Principal.....	27
147	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	28
148	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	28
149	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	28
150	4.2.4 ECP Profile Schema Usage.....	28
151	4.2.4.1 PAOS Request Header Block: SP to ECP.....	29
152	4.2.4.2 ECP Request Header Block: SP to ECP.....	30

153	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	30
154	4.2.4.4 ECP Response Header Block: IdP to ECP.....	32
155	4.2.4.5 PAOS Response Header Block: ECP to SP.....	32
156	4.2.5 Security Considerations.....	33
157	4.2.6 [E20]Use of Metadata.....	33
158	4.3 Identity Provider Discovery Profile.....	34
159	4.3.1 [E32]Required Information.....	34
160	4.3.2 Common Domain Cookie.....	34
161	4.3.3 Setting the Common Domain Cookie.....	34
162	4.3.4 Obtaining the Common Domain Cookie.....	35
163	4.4 Single Logout Profile.....	35
164	4.4.1 Required Information.....	36
165	4.4.2 Profile Overview.....	36
166	4.4.3 Profile Description.....	37
167	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	37
168	4.4.3.2 Identity Provider Determines Session Participants.....	38
169	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	38
170	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	38
171	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	39
172	4.4.4 Use of Single Logout Protocol.....	39
173	4.4.4.1 <LogoutRequest> Usage.....	39
174	4.4.4.2 <LogoutResponse> Usage.....	40
175	4.4.5 Use of Metadata.....	40
176	4.5 Name Identifier Management Profile.....	40
177	4.5.1 Required Information.....	40
178	4.5.2 Profile Overview.....	41
179	4.5.3 Profile Description.....	41
180	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	41
181	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	42
182	4.5.4 Use of Name Identifier Management Protocol.....	43
183	4.5.4.1 <ManageNameIDRequest> Usage.....	43
184	4.5.4.2 <ManageNameIDResponse> Usage.....	43
185	4.5.5 Use of Metadata.....	43
186	5 Artifact Resolution Profile.....	44
187	5.1 Required Information.....	44
188	5.2 Profile Overview.....	44
189	5.3 Profile Description.....	45
190	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	45
191	5.3.2 <ArtifactResponse> issued by Responding Entity.....	45
192	5.4 Use of Artifact Resolution Protocol.....	45
193	5.4.1 <ArtifactResolve> Usage.....	45
194	5.4.2 <ArtifactResponse> Usage.....	46
195	5.5 Use of Metadata.....	46
196	6 Assertion Query/Request Profile.....	47
197	6.1 Required Information.....	47
198	6.2 Profile Overview.....	47

199	6.3 Profile Description.....	48
200	6.3.1 Query/Request issued by SAML Requester.....	48
201	6.3.2 <Response> issued by SAML Authority.....	48
202	6.4 Use of Query/Request Protocol.....	48
203	6.4.1 Query/Request Usage.....	48
204	6.4.2 <Response> Usage.....	48
205	6.5 Use of Metadata.....	49
206	7 Name Identifier Mapping Profile.....	50
207	7.1 Required Information.....	50
208	7.2 Profile Overview.....	50
209	7.3 Profile Description.....	51
210	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	51
211	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	51
212	7.4 Use of Name Identifier Mapping Protocol.....	51
213	7.4.1 <NameIDMappingRequest> Usage.....	51
214	7.4.2 <NameIDMappingResponse> Usage.....	51
215	7.4.2.1 Limiting Use of Mapped Identifier.....	52
216	7.5 Use of Metadata.....	52
217	8 SAML Attribute Profiles.....	53
218	8.1 Basic Attribute Profile.....	53
219	8.1.1 Required Information.....	53
220	8.1.2 SAML Attribute Naming.....	53
221	8.1.2.1 Attribute Name Comparison.....	53
222	8.1.3 Profile-Specific XML Attributes.....	53
223	8.1.4 SAML Attribute Values.....	53
224	8.1.5 Example.....	53
225	8.2 X.500/LDAP Attribute Profile [E53] – Deprecated.....	53
226	8.2.1 Required Information.....	54
227	8.2.2 SAML Attribute Naming.....	54
228	8.2.2.1 Attribute Name Comparison.....	54
229	8.2.3 Profile-Specific XML Attributes.....	54
230	8.2.4 SAML Attribute Values.....	55
231	8.2.5 Profile-Specific Schema.....	56
232	8.2.6 Example.....	56
233	8.3 UUID Attribute Profile.....	56
234	8.3.1 Required Information.....	56
235	8.3.2 UUID and GUID Background.....	56
236	8.3.3 SAML Attribute Naming.....	57
237	8.3.3.1 Attribute Name Comparison.....	57
238	8.3.4 Profile-Specific XML Attributes.....	57
239	8.3.5 SAML Attribute Values.....	57
240	8.3.6 Example.....	57
241	8.4 DCE PAC Attribute Profile.....	58
242	8.4.1 Required Information.....	58

243	8.4.2 PAC Description.....	58
244	8.4.3 SAML Attribute Naming.....	58
245	8.4.3.1 Attribute Name Comparison.....	58
246	8.4.4 Profile-Specific XML Attributes.....	59
247	8.4.5 SAML Attribute Values.....	59
248	8.4.6 Attribute Definitions.....	59
249	8.4.6.1 Realm.....	60
250	8.4.6.2 Principal.....	60
251	8.4.6.3 Primary Group.....	60
252	8.4.6.4 Groups.....	60
253	8.4.6.5 Foreign Groups.....	60
254	8.4.7 Example.....	61
255	8.5 XACML Attribute Profile.....	62
256	8.5.1 Required Information.....	62
257	8.5.2 SAML Attribute Naming.....	62
258	8.5.2.1 Attribute Name Comparison.....	62
259	8.5.3 Profile-Specific XML Attributes.....	62
260	8.5.4 SAML Attribute Values.....	62
261	8.5.5 Profile-Specific Schema.....	63
262	8.5.6 Example.....	63
263	9 References.....	64
264	Appendix A. Acknowledgments.....	67
265	Appendix B. Notices.....	69

1 Introduction

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

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

1.1 Profile Concepts

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

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

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

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

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

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

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

1.2 Notation

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

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

308 described in IETF RFC 2119 [RFC2119].

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

310 Example code listings appear like this.

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

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

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
md:	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace [SAMLMeta].
ecp:	urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp	This is the SAML V2.0 ECP profile namespace, specified in this document and in a schema [SAMLECP-xsd].
ds:	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
xenc:	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
SOAP-ENV:	http://schemas.xmlsoap.org/soap/envelope	This is the SOAP V1.1 namespace [SOAP1.1].
paos:	urn:liberty:paos:2003-08	This is the Liberty Alliance PAOS namespace.
dce:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE	This is the SAML V2.0 DCE PAC attribute profile namespace, specified in this document and in a schema [SAMLDCExsd].
x500:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500	This is the SAML V2.0 X.500/LDAP attribute profile namespace, specified in this document and in a schema [SAMLX500-xsd].
xacmlprof:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML	This is the SAML V2.0 XACML attribute profile namespace, specified in this document and in a schema [SAMLXAC-xsd].
[E71]xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.

Prefix	XML Namespace	Comments
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

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

2 Specification of Additional Profiles

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

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

2.1 Guidelines for Specifying Profiles

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

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

2.2 Guidelines for Specifying Attribute Profiles

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

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

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

3 Confirmation Method Identifiers

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

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

It is anticipated that profiles will define and use several different values for `[E56]ConfirmationMethod`, each corresponding to a different SAML usage scenario. The following methods are defined for use by profiles defined within this specification and other profiles that find them useful.

3.1 Holder of Key

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

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

As described in [XMLSig], each `<ds:KeyInfo>` element holds a key or information that enables an application to obtain a key. The holder of `[E47]one or more of the specified keysa-specified-key` is considered to be `[E40]an acceptable attesting entity forthe-subject-of` the assertion by the asserting party.

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

`[E47]If the keys contained in the <SubjectConfirmationData> element belong to an entity other than the subject, then the asserting party SHOULD identify that entity to the relying party by including a SAML identifier representing it in the enclosing <SubjectConfirmation> element.`

`Note that a given <SubjectConfirmation> element using the Holder of Key method SHOULD include keys belonging to only a single attesting entity. If multiple attesting entities are to be permitted to use the assertion, then multiple <SubjectConfirmation> elements SHOULD be included.`

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

```
<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
  <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
    <ds:KeyInfo>
      <ds:KeyName>By-Tor</ds:KeyName>
    </ds:KeyInfo>
    <ds:KeyInfo>
      <ds:KeyName>Snow Dog</ds:KeyName>
    </ds:KeyInfo>
  </SubjectConfirmationData>
</SubjectConfirmation>
```

405 </SubjectConfirmation>

406 3.2 Sender Vouches

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

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

412 3.3 Bearer

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

414 The subject of the assertion is [E47]the bearer of considered to be an acceptable attesting entity for the
415 assertion by the asserting party, subject to optional constraints on confirmation using the attributes that
416 MAY be present in the <SubjectConfirmationData> element, as defined by [SAMLCore].

417 If the intended bearer is known by the asserting party to be an entity other than the subject, then the
418 asserting party SHOULD identify that entity to the relying party by including a SAML identifier
419 representing it in the enclosing <SubjectConfirmation> element.

420 If multiple attesting entities are to be permitted to use the assertion based on bearer semantics, then
421 multiple <SubjectConfirmation> elements SHOULD be included.

422 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
423 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
424 19th, 2004, in response to a request with ID "_1234567890".

```
425 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
426   <SubjectConfirmationData InResponseTo="_1234567890"  
427     Recipient="https://www.serviceprovider.com/saml/consumer"  
428     NotOnOrAfter="2004-03-19T13:27:00Z"  
429   </SubjectConfirmationData>  
430 </SubjectConfirmation>
```

4 SSO Profiles of SAML

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

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

4.1 Web Browser SSO Profile

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

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

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

4.1.1 Required Information

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

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

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

Description: Given below.

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

4.1.2 Profile Overview

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

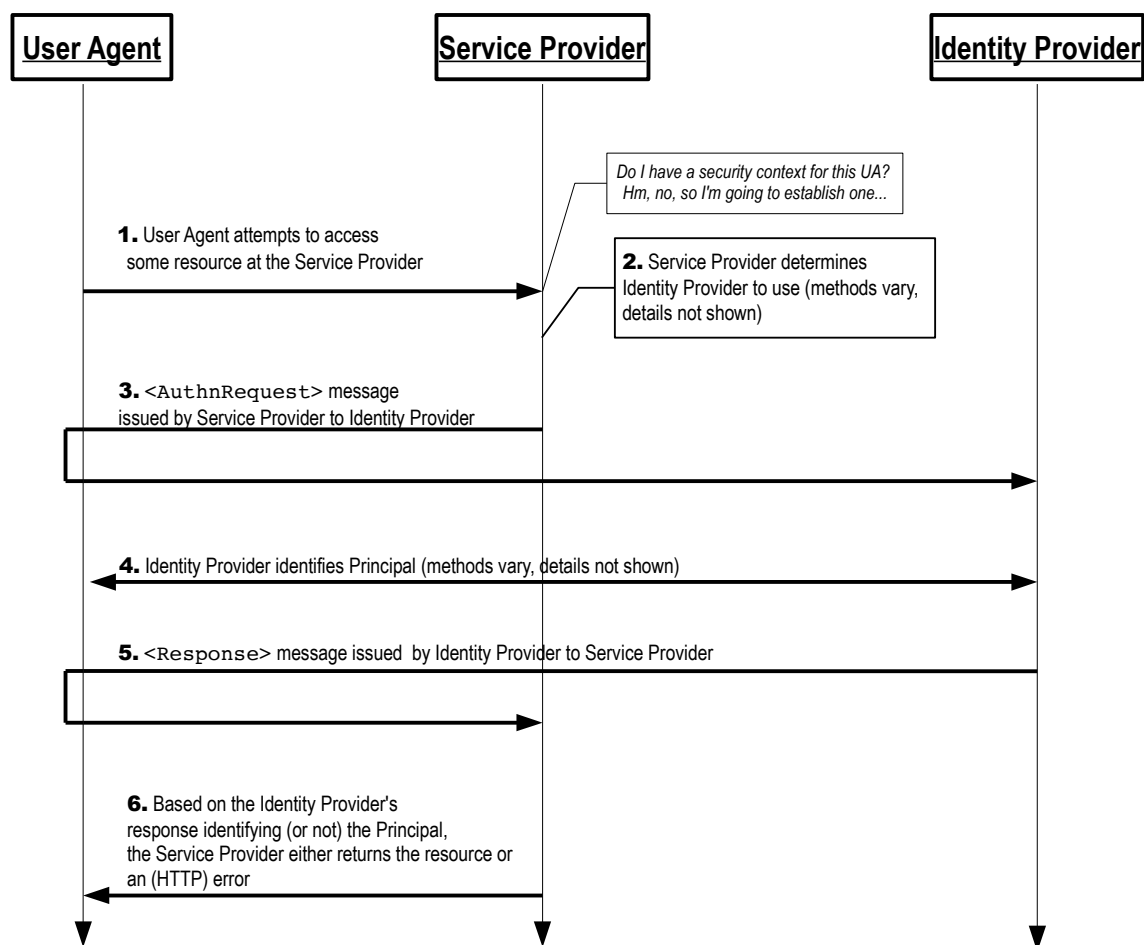


Figure 1

463 1. HTTP Request to Service Provider

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

466 2. Service Provider Determines Identity Provider

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

471 3. <AuthnRequest> issued by Service Provider to Identity Provider

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

475 4. Identity Provider identifies Principal

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

5. Identity Provider issues <Response> to Service Provider

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

6. Service Provider grants or denies access to Principal

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

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

4.1.3 Profile Description

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

Single Sign-On Service

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

Assertion Consumer Service

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

4.1.3.1 HTTP Request to Service Provider

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

4.1.3.2 Service Provider Determines Identity Provider

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

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

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

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

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

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

4.1.3.4 Identity Provider Identifies Principal

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

4.1.3.5 Identity Provider Issues <Response> to Service Provider

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

[E85] Identity provider implementations SHOULD support the issuance of <saml2p:Response> messages (with appropriate status codes) in the event of an error condition, provided that the user agent remains available and an acceptable location to which to deliver the response is available. The criteria for "acceptability" of a response location are not formally specified, but are subject to identity provider policy and reflect its responsibility to protect users from being sent to untrusted or possibly malicious parties.

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

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

It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 [SSL3] or TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The <Assertion> element(s) in the <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP Artifact binding is used. [E93] For the purposes of the profile, either the <Response> or the <Assertion> element(s) in the <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-Artifact binding is used. If an <EncryptedAssertion> element is present

and a CBC-mode algorithm is used, then the <Response> SHOULD be signed to ensure the ciphertext is integrity protected (see section 6.2 of [SAMLCore]).

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

4.1.3.6 Service Provider Grants or Denies Access to User Agent

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

4.1.4 Use of Authentication Request Protocol

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

4.1.4.1 <AuthnRequest> Usage

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

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

~~[E14]If the service provider wishes to permit the identity provider to establish a new identifier for the principal if none exists, it MUST include a <NameIDPolicy> element with the AllowCreate attribute set to "true". Otherwise, only a principal for whom the identity provider has previously established an identifier usable by the service provider can be authenticated successfully. This profile does not provide any guidelines for the use of AllowCreate; see [SAMLCore] for normative rules on using AllowCreate.~~

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

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

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

4.1.4.2 <Response> Usage

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

- [E17]The <Issuer> element MAY be omitted, but if present If the <Response> message is signed or if an enclosed assertion is encrypted, then the <Issuer> element MUST be present. Otherwise it MAY be omitted. If present it MUST contain the unique identifier of the issuing identity provider; the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the unique identifier of the [E26]issuingresponding identity provider; the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity. Note that this profile assumes a single responding identity provider, and all assertions in a response MUST be issued by the same entity.
- The set of one or more assertions MUST contain at least one <AuthnStatement> that reflects the authentication of the principal to the identity provider.
- At least one assertion containing an <AuthnStatement> MUST contain a <Subject> element with at least one <SubjectConfirmation> element containing a Method of urn:oasis:names:tc:SAML:2.0:cm:bearer. If the identity provider supports the Single Logout profile, defined in Section 4.4, any such authentication statements MUST include a SessionIndex attribute to enable per-session logout requests by the service provider.
- The bearer <SubjectConfirmation> element described above MUST contain a <SubjectConfirmationData> element that contains a Recipient attribute containing the service provider's assertion consumer service URL and a NotOnOrAfter attribute that limits the window during which the assertion can be delivered. It MAY contain an Address attribute limiting the client address from which the assertion can be delivered. It MUST NOT contain a NotBefore attribute. If the containing message is in response to an <AuthnRequest>, then the InResponseTo attribute MUST match the request's ID. If multiple assertions are included, then each assertion's <Subject> element MUST refer to the same principal. It is allowable for the content of the <Subject> elements to differ (e.g. using different <NameID> or alternative <SubjectConfirmation> elements).
- Any assertion issued for consumption using this profile MUST contain a <Subject> element with at least one <SubjectConfirmation> element containing a Method of urn:oasis:names:tc:SAML:2.0:cm:bearer. Such an assertion is termed a bearer assertion. Bearer assertions MAY contain additional <SubjectConfirmation> elements.
- Assertions without a bearer <SubjectConfirmation> MAY also be included; processing of additional assertions or <SubjectConfirmation> elements is outside the scope of this profile.
- At lease one bearer <SubjectConfirmation> element MUST contain a <SubjectConfirmationData> element that itself MUST contain a Recipient attribute containing the service provider's assertion consumer service URL and a NotOnOrAfter attribute that limits the window during which the assertion can be [E52]confirmed by the relying party. It MAY also contain an Address attribute limiting the client address from which the assertion can be delivered. It MUST NOT contain a NotBefore attribute. If the containing message is in response to an <AuthnRequest>, then the InResponseTo attribute MUST match the request's ID.
- The set of one or more bearer assertions MUST contain at least one <AuthnStatement> that reflects the authentication of the principal to the identity provider. Multiple <AuthnStatement> elements MAY be included, but the semantics of multiple statements is not defined by this profile.
- If the identity provider supports the Single Logout profile, defined in Section 4.4, any authentication statements MUST include a SessionIndex attribute to enable per-session logout requests by the service provider.

- Other statements ~~and confirmation methods~~ MAY be included in the [bearer](#) assertion(s) at the discretion of the identity provider. In particular, <AttributeStatement> elements MAY be included. The <AuthnRequest> MAY contain an AttributeConsumingServiceIndex XML attribute referencing information about desired or required attributes in [SAMLMeta]. The identity provider MAY ignore this, or send other attributes at its discretion.
- ~~TheEach bearer~~ assertion(s) ~~containing a bearer subject confirmation~~ MUST contain an <AudienceRestriction> including the service provider's unique identifier as an <Audience>.
- Other conditions (and other <Audience> elements) MAY be included as requested by the service provider or at the discretion of the identity provider. (Of course, all such conditions MUST be understood by and accepted by the service provider in order for the assertion to be considered valid.)
- The identity provider is NOT obligated to honor the requested set of <Conditions> in the <AuthnRequest>, if any.

4.1.4.3 <Response> Message Processing Rules

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

- Verify any signatures present on the assertion(s) or the response
- Verify that the Recipient attribute in [\[E26\]theany](#) bearer <SubjectConfirmationData> matches the assertion consumer service URL to which the <Response> or artifact was delivered
- Verify that the NotOnOrAfter attribute in [theany](#) bearer <SubjectConfirmationData> has not passed, subject to allowable clock skew between the providers
- Verify that the InResponseTo attribute in the bearer <SubjectConfirmationData> equals the ID of its original <AuthnRequest> message, unless the response is unsolicited (see Section 4.1.5), in which case the attribute MUST NOT be present
- Verify that any assertions relied upon are valid in other respects. [Note that while multiple bearer <SubjectConfirmation> elements may be present, the successful evaluation of a single such element in accordance with this profile is sufficient to confirm an assertion. However, each assertion, if more than one is present, MUST be evaluated independently.](#)
- If [anythe](#) bearer <SubjectConfirmationData> includes an Address attribute, the service provider MAY check the user agent's client address against it.
- Any assertion which is not valid, or whose subject confirmation requirements cannot be met SHOULD be discarded and SHOULD NOT be used to establish a security context for the principal.
- If an <AuthnStatement> used to establish a security context for the principal contains a SessionNotOnOrAfter attribute, the security context SHOULD be discarded once this time is reached, unless the service provider reestablishes the principal's identity by repeating the use of this profile. [Note that if multiple <AuthnStatement> elements are present, the SessionNotOnOrAfter value closest to the present time SHOULD be honored.](#)

[\[E93\] Note that if <EncryptedAssertion> elements are present and a CBC-mode algorithm is used, then the <Response> SHOULD be signed to ensure the ciphertext is integrity protected \(see section 6.2 of \[SAMLCore\]\). Some deployments may require both the <Response> and any <Assertion> elements be signed to address both the encryption issue and non-repudiation of the assertion \(the latter being outside the scope of SAML\).](#)

4.1.4.4 Artifact-Specific <Response> Message Processing Rules

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

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

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

4.1.4.5 POST-Specific Processing Rules

If the HTTP POST binding is used to deliver the <Response>, ~~[E26]the enclosed assertion(s) MUST be signed. Each assertion MUST be protected by a digital signature. This can be accomplished by signing each individual <Assertion> element or by signing the <Response> element.~~

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

4.1.5 Unsolicited Responses

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

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

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

[E90] Note that the use of unsolicited responses can lead to Cross-Site Request Forgery (CSRF) vulnerabilities due to the inability to ensure that a request from the client originated the SAML profile transaction. Service providers SHOULD have a means of disabling the acceptance of unsolicited responses if circumstances warrant. The use of solicited responses may also be vulnerable to such attacks, the use of cookies to correlate the issuance of SAML requests and responses with the same client being one possible solution. However, if unsolicited responses cannot be prevented, no improvement to the solicited case will be of use.

4.1.6 [E90] Use of Relay State

The RelayState feature of the various HTTP-based bindings defined for use with this profile MAY be used to preserve information about resources requested by the user agent prior to the use of the profile. As discussed in Error: Reference source not found, the lack of integrity protection in many scenarios, including the case of unsolicited responses, makes it essential for identity and service providers to perform appropriate sanitization of the RelayState value and any URLs derived from it. The URL scheme eventually derived SHOULD be limited to "https" or "http", and protection against unencoded executable content must be applied.

4.1.7 Use of Metadata

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

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

The providers MAY document the key(s) used to sign requests, responses, and assertions with `<md:KeyDescriptor>` elements with a `use` attribute of [\[E58\]signing](#). When encrypting SAML elements, `<md:KeyDescriptor>` elements with a `use` attribute of [encryption](#) MAY be used to document supported encryption algorithms and settings, and public keys used to receive bulk encryption keys.

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

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

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

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

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

4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

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

4.2.1 Required Information

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

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

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

Description: Given below.

Updates: None.

4.2.2 Profile Overview

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

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

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

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

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

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

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

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

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

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

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

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

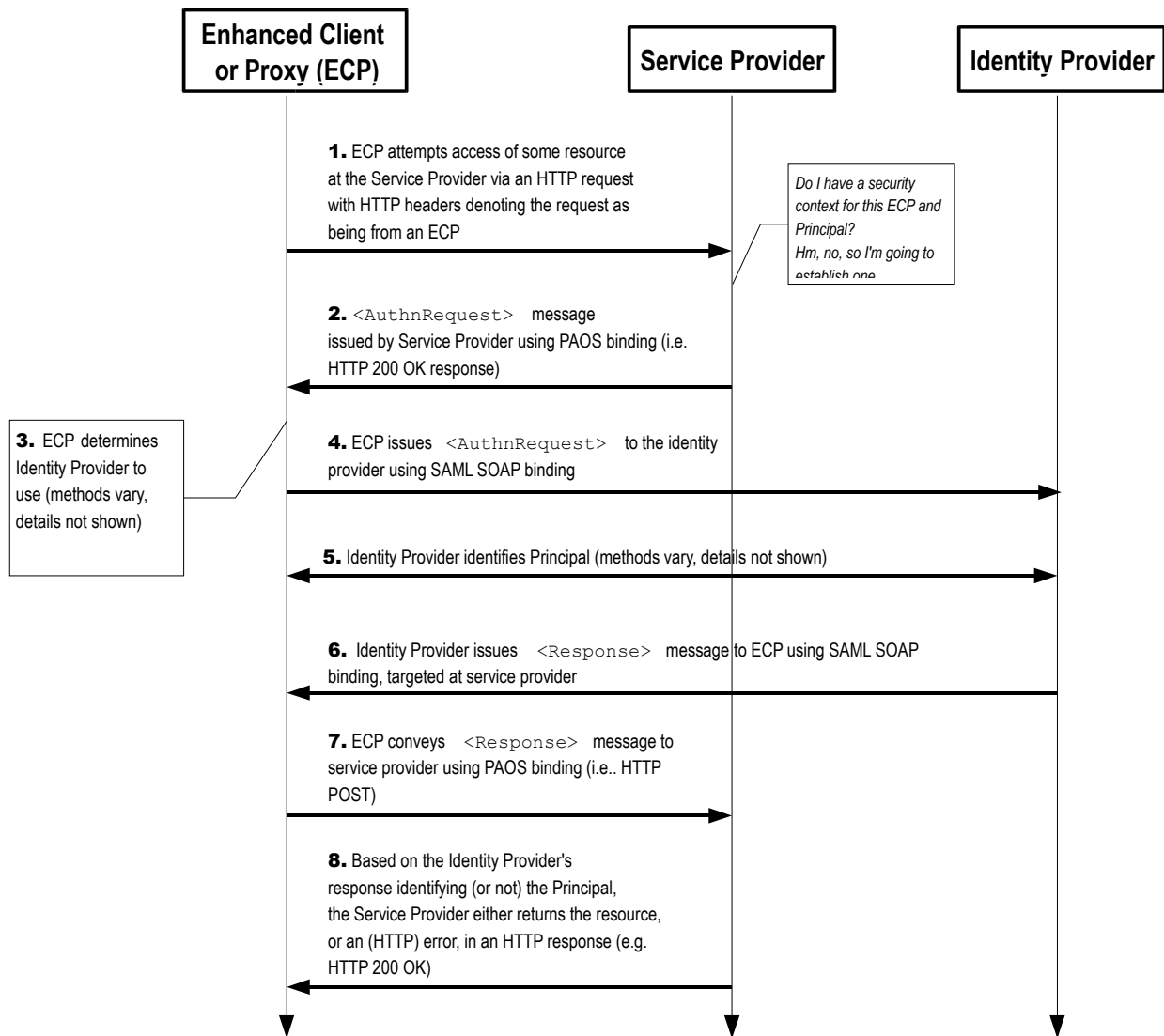


Figure 2

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

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

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

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

842 In step 2, the service provider issues an <AuthnRequest> message to the ECP, which is to be
843 delivered by the ECP to the appropriate identity provider. The Reverse SOAP (PAOS) binding

844 [SAMLBind] is used here.

845 3. ECP Determines Identity Provider

846 In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication
847 request protocol that supports its preferred binding. The means by which this is accomplished is
848 implementation-dependent. ~~[E18] The ECP MAY use the SAML identity provider discovery profile~~
849 ~~described in Section 4.3.~~

850 4. ECP conveys <AuthnRequest> to Identity Provider

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

855 5. Identity Provider identifies Principal

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

859 6. Identity Provider issues <Response> to ECP, targeted at Service Provider

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

863 7. ECP conveys <Response> message to Service Provider

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

866 8. Service Provider grants or denies access to Principal

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

870 4.2.3 Profile Description

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

872 4.2.3.1 ECP issues HTTP Request to Service Provider

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

- 875 1. The HTTP Accept Header field indicating the ability to accept the MIME type
876 "application/vnd.paos+xml"
- 877 2. The HTTP PAOS Header field specifying the PAOS version with urn:liberty:paos:2003-08 at
878 minimum.
- 879 3. Furthermore, support for this profile MUST be specified in the HTTP PAOS Header field as a service
880 value, with the value [E54] "urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp". This value
881 should correspond to the service attribute in the PAOS Request SOAP header block

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

883 GET /index HTTP/1.1

```

884 Host: identity-service.example.com
885 Accept: text/html; application/vnd.paos+xml
886 PAOS: ver="urn:liberty:paos:2003-08" ;
887 urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp

```

888 4.2.3.2 Service Provider Issues <AuthnRequest> to ECP

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

893 The SOAP envelope MUST contain:

- 894 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the
895 identity provider.
- 896 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of
897 http://schemas.xmlsoap.org/soap/actor/next. This header block provides control
898 information such as the URL to which to send the response in this solicit-response message
899 exchange pattern.
- 900 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
901 http://schemas.xmlsoap.org/soap/actor/next. The ECP Request header block defines
902 information related to the authentication request that the ECP may need to process it, such as a list
903 of identity providers acceptable to the service provider, whether the ECP may interact with the
904 principal through the client, and the service provider's human-readable name that may be displayed
905 to the principal.

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

909 4.2.3.3 ECP Determines Identity Provider

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

911 4.2.3.4 ECP issues <AuthnRequest> to Identity Provider

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

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

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

923 4.2.3.5 Identity Provider Identifies Principal

924 At any time during the previous step or subsequent to it, the identity provider MUST establish the identity
925 of the principal (unless it returns an error to the service provider). The ForceAuthn <AuthnRequest>
926 attribute, if present with a value of true, obligates the identity provider to freshly establish this identity,

rather than relying on an existing session it may have with the principal. Otherwise, and in all other respects, the identity provider may use any means to authenticate the user agent, subject to any requirements included in the <AuthnRequest> in the form of the <RequestedAuthnContext> element.

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

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

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

4.2.3.7 ECP Conveys <Response> Message to Service Provider

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

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

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

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

4.2.3.8 Service Provider Grants or Denies Access to Principal

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

4.2.4 ECP Profile Schema Usage

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

```
<schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
```

```

971   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
972   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
973   elementFormDefault="unqualified"
974   attributeFormDefault="unqualified"
975   blockDefault="substitution"
976   version="2.0">
977   <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
978     schemaLocation="saml-schema-protocol-2.0.xsd"/>
979   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
980     schemaLocation="saml-schema-assertion-2.0.xsd"/>
981   <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
982     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
983   <annotation>
984     <documentation>
985       Document identifier: saml-schema-ecp-2.0
986       Location: http://docs.oasis-open.org/security/saml/v2.0/
987       Revision history:
988         V2.0 (March, 2005):
989         Custom schema for ECP profile, first published in SAML 2.0.
990     </documentation>
991   </annotation>

992   <element name="Request" type="ecp:RequestType"/>
993   <complexType name="RequestType">
994     <sequence>
995       <element ref="saml:Issuer"/>
996       <element ref="samlp:IDPList" minOccurs="0"/>
997     </sequence>
998     <attribute ref="S:mustUnderstand" use="required"/>
999     <attribute ref="S:actor" use="required"/>
1000    <attribute name="ProviderName" type="string" use="optional"/>
1001    <attribute name="IsPassive" type="boolean" use="optional"/>
1002  </complexType>

1003
1004  <element name="Response" type="ecp:ResponseType"/>
1005  <complexType name="ResponseType">
1006    <attribute ref="S:mustUnderstand" use="required"/>
1007    <attribute ref="S:actor" use="required"/>
1008    <attribute name="AssertionConsumerServiceURL" type="anyURI"
1009  use="required"/>
1010  </complexType>

1011
1012  <element name="RelayState" type="ecp:RelayStateType"/>
1013  <complexType name="RelayStateType">
1014    <simpleContent>
1015      <extension base="string">
1016        <attribute ref="S:mustUnderstand" use="required"/>
1017        <attribute ref="S:actor" use="required"/>
1018      </extension>
1019    </simpleContent>
1020  </complexType>
1021 </schema>

```

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

4.2.4.1 PAOS Request Header Block: SP to ECP

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

responseConsumerURL [Required]

Specifies where the ECP is to send an error response. Also used to verify the correctness of the

1028 identity provider's response, by cross checking this location against the
 1029 AssertionServiceConsumerURL in the ECP response header block. This value MUST be the
 1030 same as the ~~[E22]AssertionServiceConsumerURL~~ AssertionConsumerServiceURL (or the
 1031 URL referenced in metadata) conveyed in the <AuthnRequest> ~~[E35]~~ and SHOULD NOT be a
 1032 relative URL.

1033 service [Required]
 1034 Indicates that the PAOS service being used is this SAML authentication profile. The value MUST be
 1035 urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp.

1036 SOAP-ENV:mustUnderstand [Required]
 1037 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
 1038 understood.

1039 SOAP-ENV:actor [Required]
 1040 The value MUST be http://schemas.xmlsoap.org/soap/actor/next.

1041 messageID [Optional]
 1042 Allows optional response correlation. It MAY be used in this profile, but is NOT required, since this
 1043 functionality is provided by the SAML protocol layer, via the ID attribute in the <AuthnRequest> and
 1044 the InResponseTo attribute in the <Response>.

1045 The PAOS Request SOAP header block has no element content.

1046 **4.2.4.2 ECP Request Header Block: SP to ECP**

1047 The ECP Request SOAP header block is used to convey information needed by the ECP to process the
 1048 authentication request. It is mandatory and its presence signals the use of this profile. It contains the
 1049 following elements and attributes:

1050 SOAP-ENV:mustUnderstand [Required]
 1051 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not
 1052 understood.

1053 SOAP-ENV:actor [Required]
 1054 The value MUST be http://schemas.xmlsoap.org/soap/actor/next.

1055 ProviderName [Optional]
 1056 A human-readable name for the requesting service provider.

1057 IsPassive [Optional]
 1058 A boolean value. If true, the identity provider and the client itself MUST NOT take control of the user
 1059 interface from the request issuer and interact with the principal in a noticeable fashion. If a value is
 1060 not provided, the default is true.

1061 <saml:Issuer> [Required]
 1062 This element MUST contain the unique identifier of the requesting service provider; the Format
 1063 attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
 1064 format:entity.

1065 <samlp:IDPList> [Optional]
 1066 Optional list of identity providers that the service provider recognizes and from which the ECP may
 1067 choose to service the request. See [SAMLCore] for details on the content of this element.

4.2.4.3 ECP RelayState Header Block: SP to ECP

The ECP RelayState SOAP header block is used to convey state information from the service provider that it will need later when processing the response from the ECP. It is optional, but if used, the ECP MUST include an identical header block in the response in step [E27]57. It contains the following attributes:

SOAP-ENV:mustUnderstand [Required]

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

SOAP-ENV:actor [Required]

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

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

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

```
<SOAP-ENV:Envelope
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
      responseConsumerURL="[E35]http://identity-
service.example.com/abchttps://ServiceProvider.example.com/ecp_assertion_consumer"
      messageID="6c3a4f8b9c2d" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
ENV:mustUnderstand="1"
      service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
    </paos:Request>
    <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
      SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
      ProviderName="Service Provider X" IsPassive="0">
    <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
    <samlp:IDPList>
      <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
        Name="Identity Provider X"
        Loc="https://IdentityProvider.example.com/saml2/sso"
      </samlp:IDPEntry>
      <samlp:GetComplete>
        https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
      </samlp:GetComplete>
    </samlp:IDPList>
    </ecp:Request>
    <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
      SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
    ...
    </ecp:RelayState>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <samlp:AuthnRequest> ... </samlp:AuthnRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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

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

4.2.4.4 ECP Response Header Block: IdP to ECP

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

SOAP-ENV:mustUnderstand [Required]

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

SOAP-ENV:actor [Required]

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

AssertionConsumerServiceURL [Required]

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

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

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

The ECP Response SOAP header has no element content.

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

```
<SOAP-ENV:Envelope
  xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion
consumer"/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <samlp:Response> ... </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.2.4.5 PAOS Response Header Block: ECP to SP

The PAOS Response header block includes the following attributes:

SOAP-ENV:mustUnderstand [Required]

1168 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
 1169 understood.

1170 SOAP-ENV:actor [Required]

1171 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1172 refToMessageID [Optional]

1173 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
 1174 MUST be added by the ECP if the corresponding PAOS request specified the `messageID` attribute.
 1175 Note that the equivalent functionality is provided in SAML using `<AuthnRequest>` and `<Response>`
 1176 correlation.

1177 The PAOS Response SOAP header has no element content.

1178 Following is an example of an ECP-to-SP response.

```

1179 <SOAP-ENV:Envelope
1180     xmlns:paos="urn:liberty:paos:2003-08"
1181     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1182     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1183   <SOAP-ENV:Header>
1184     <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1185     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1186     ENV:mustUnderstand="1"/>
1187     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1188     SOAP-ENV:mustUnderstand="1" SOAP-
1189     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1190       ...
1191     </ecp:RelayState>
1192   </SOAP-ENV:Header>
1193   <SOAP-ENV:Body>
1194     <samlp:Response> ... </samlp:Response>
1195   </SOAP-ENV:Body>
1196 </SOAP-ENV:Envelope>
  
```

1197 4.2.5 Security Considerations

1198 The `<AuthnRequest>` message SHOULD be signed. Per the rules specified by the browser SSO profile,
 1199 the assertions enclosed in the `<Response>` [\[E93\] or the `<Response>` itself](#), MUST be signed. The
 1200 delivery of the response in the SOAP envelope via PAOS is essentially analogous to the use of the HTTP
 1201 POST binding and security countermeasures appropriate to that binding are used.

1202 [\[E93\] Note that if `<EncryptedAssertion>` elements are present and a CBC-mode algorithm is used,](#)
 1203 [then the `<Response>` SHOULD be signed to ensure the ciphertext is integrity protected \(see section 6.2](#)
 1204 [of \[SAMLCore\]\). Some deployments may require both the `<Response>` and any `<Assertion>` elements](#)
 1205 [be signed to address both the encryption issue and non-repudiation of the assertion \(the latter being](#)
 1206 [outside the scope of SAML\).](#)

1207 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the
 1208 use of SSL/TLS over every HTTP exchange with the client.

1209 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS
 1210 authentication.

1211 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
 1212 session. Any HTTP exchanges subsequent to the delivery of the `<AuthnRequest>` message and before
 1213 the identity provider returns a `<Response>` MUST be securely associated with the original request.

1214 [\[E90\] The `RelayState` header block defined for use with this profile MAY be used to preserve information](#)
 1215 [about resources requested by the client prior to the use of the profile. As discussed in \[SAMLBind\], the](#)

1216 [lack of integrity protection in many scenarios, including the case of unsolicited responses, makes it](#)
1217 [essential for identity and service providers to perform appropriate sanitization of the RelayState value and](#)
1218 [any URLs derived from it. The URL scheme eventually derived SHOULD be limited to "https" or "http",](#)
1219 [and protection against unencoded executable content must be applied.](#)

1220 **4.2.6 [E20]Use of Metadata**

1221 [The rules specified in the browser SSO profile in Section 4.1.7 apply here as well. Specifically, the](#)
1222 [indexed endpoint element <md:AssertionConsumerService> with a binding of](#)
1223 [urn:oasis:names:tc:SAML:2.0:bindings:PAOS MAY be used to describe the supported binding](#)
1224 [and location\(s\) to which an identity provider may send responses to a service provider using this profile.](#)
1225 [IN addition, the endpoint <md:SingleSignOnService> with a binding of](#)
1226 [urn:oasis:names:tc:SAML:2.0:bindings:SOAP MAY be used to describe the supported binding](#)
1227 [and location\(s\) to which an service provider may send requests to an identity provider using this profile.](#)

1228 **4.3 Identity Provider Discovery Profile**

1229 This section defines a profile by which a service provider can discover which identity providers a principal
1230 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
1231 service providers need a means to discover which identity provider(s) a principal uses. The discovery
1232 profile relies on a cookie that is written in a domain that is common between identity providers and service
1233 providers in a deployment. The domain that the deployment predetermines is known as the common
1234 domain in this profile, and the cookie containing the list of identity providers is known as the common
1235 domain cookie.

1236 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
1237 this profile.

1238 **4.3.1 [E32]Required Information**

1239 **Identification:** [urn:oasis:names:tc:SAML:2.0:profiles:SSO:idp-discovery](#)

1240 **Contact information:** [security-services-comment@lists.oasis-open.org](#)

1241 **Description:** [Given below.](#)

1242 **Updates:** [None.](#)

1243 **4.3.2 Common Domain Cookie**

1244 The name of the cookie MUST be "_saml_idp". The format of the cookie value MUST be a set of one or
1245 more base-64 encoded URI values separated by a single space character. Each URI is the unique
1246 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
1247 URL encoded.

1248 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
1249 identifier to the list. If the identifier is already present in the list, it MAY remove and append it. The intent is
1250 that the most recently established identity provider session is the last one in the list.

1251 The cookie MUST be set with a Path prefix of "/". The Domain MUST be set to "[common-domain]" where
1252 [common-domain] is the common domain established within the deployment for use with this profile.
1253 There MUST be a leading period. The cookie MUST be marked as secure.

1254 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY
1255 be either session-only or persistent. This choice may be made within a deployment, but should apply
1256 uniformly to all identity providers in the deployment. [\[E63\]Note that while a session-only cookie can be](#)
1257 [used, the intent of this profile is not to provide a means of determining whether a user actually has an](#)

active session with one or more of the identity providers stored in the cookie. The cookie merely identifies identity providers known to have been used in the past. Service providers MAY instead rely on the `IsPassive` attribute in their `<samlp:AuthnRequest>` message to probe for active sessions.

4.3.3 Setting the Common Domain Cookie

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

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

4.3.4 Obtaining the Common Domain Cookie

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

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

The specific means by which the service provider reads the cookie are implementation-specific so long as it is able to cause the user agent to present cookies that have been set with the parameters given in Section 4.3.2. One possible implementation strategy is described as follows and should be considered non-normative. Additionally, it may be sub-optimal for some applications.

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

4.4 Single Logout Profile

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

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

Note that a principal (or an administrator terminating a principal's session) may choose to terminate this "global" session either by contacting the session authority, or an individual session participant. Also note

1301 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1302 which it is the relying party for another identity provider's assertions regarding that principal.

1303 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1304 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1305 front-channel binding may be required, for example, in cases in which a principal's session state exists
1306 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1307 session participant or session authority is required. As will be discussed below, session participants
1308 should if possible use a "front-channel" binding when initiating this profile to maximize the likelihood that
1309 the session authority can propagate the logout successfully to all participants.

1310 **4.4.1 Required Information**

1311 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout
1312 **Contact information:** security-services-comment@lists.oasis-open.org
1313 **Description:** Given below.
1314 **Updates:** None

1315 **4.4.2 Profile Overview**

1316 Figure 3 illustrates the basic template for achieving single logout:

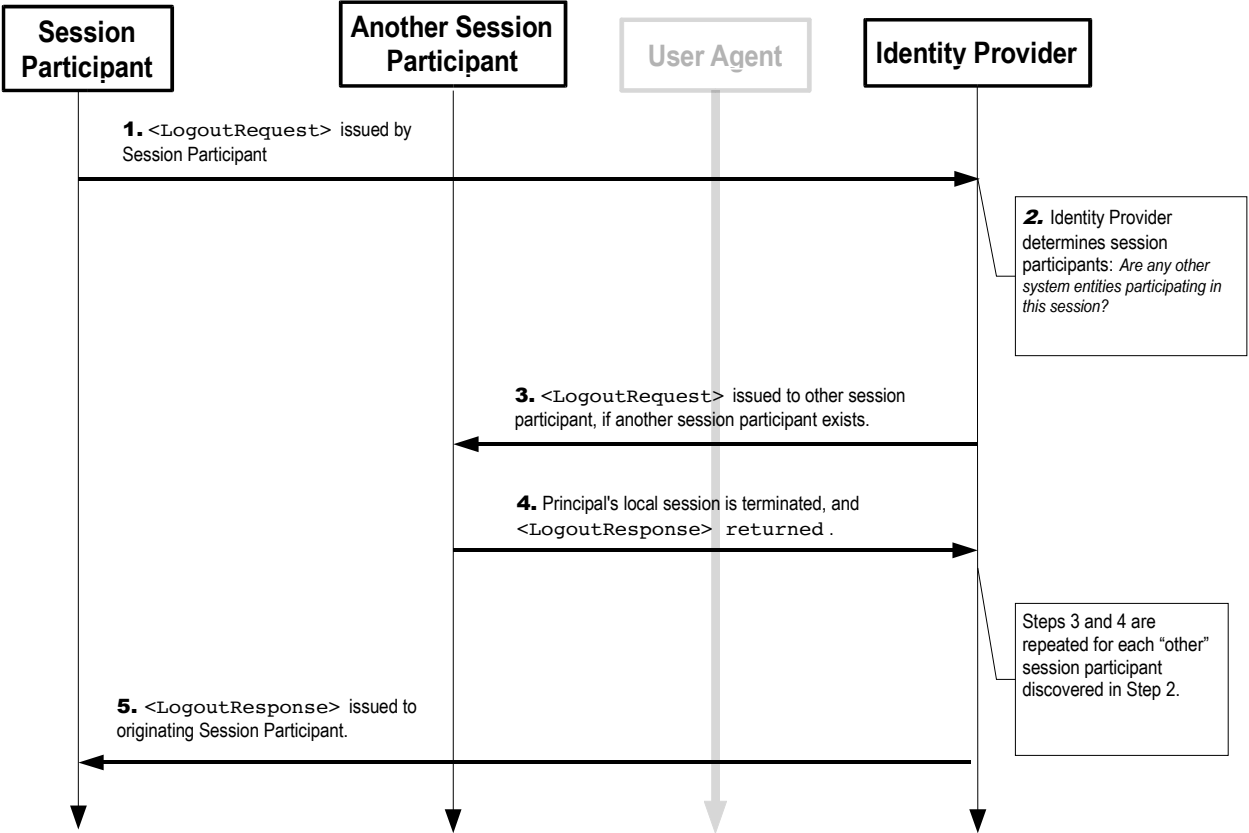


Figure 3

1317 The grayed-out user agent illustrates that the message exchange may pass through the user agent or

1318 may be a direct exchange between system entities, depending on the SAML binding used to implement
1319 the profile.

1320 The following steps are described by the profile. Within an individual step, there may be one or more
1321 actual message exchanges depending on the binding used for that step and other implementation-
1322 dependent behavior.

1323 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1324 In step 1, the session participant initiates single logout and terminates a principal's session(s) by
1325 sending a <LogoutRequest> message to the identity provider from whom it received the
1326 corresponding authentication assertion. The request may be sent directly to the identity provider
1327 or sent indirectly through the user agent.

1328 **2. Identity Provider determines Session Participants**

1329 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if
1330 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
1331 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
1332 are repeated for each session participant identified.

1333 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1334 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or
1335 session authority related to one or more of the session(s) being terminated. The request may be
1336 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of
1337 the request in step 1).

1338 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1339 In step 4, a session participant or session authority terminates the principal's session(s) as
1340 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.
1341 The response may be returned directly to the identity provider or indirectly through the user agent
1342 (if consistent with the form of the request in step 3).

1343 **5. Identity Provider issues <LogoutResponse> to Session Participant**

1344 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting
1345 session participant. The response may be returned directly to the session participant or indirectly
1346 through the user agent (if consistent with the form of the request in step 1).

1347 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1348 <LogoutRequest> to all session participants, also skipping step 5.

1349 **4.4.3 Profile Description**

1350 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1351 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1352 **Single Logout Service**

1353 This is the single logout protocol endpoint at an identity provider or session participant to which the
1354 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are
1355 delivered. The same or different endpoints MAY be used for requests and responses.

4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider

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

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

Asynchronous Bindings (Front-Channel)

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

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

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

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

Synchronous Bindings (Back-Channel)

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

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

4.4.3.2 Identity Provider Determines Session Participants

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

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

4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority

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

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

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

4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider

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

Synchronous Bindings (Back-Channel)

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

Asynchronous Bindings (Front-Channel)

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

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

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

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

4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant

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

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

1444 both entities MAY be used.

1445 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1446 4.4.4.2.

1447 4.4.4 Use of Single Logout Protocol

1448 4.4.4.1 <LogoutRequest> Usage

1449 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting
1450 entity; the Format attribute MUST be omitted or have a value of
1451 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1452 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1453 the message or using a binding-specific mechanism.

1454 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1455 the authentication assertion the requester issued or received regarding the session being terminated, per
1456 the matching rules defined in Section 3.3.4 of [SAMLCore].

1457 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the
1458 request. [E38](Note that the session participant always receives a SessionIndex attribute in the
1459 <saml:AuthnStatement> elements that it receives to initiate the session, per Section 4.1.4.2 of the
1460 Web Browser SSO Profile.) If the requester is a session authority (or acting on its behalf), then it MAY
1461 omit any such elements to indicate the termination of all of the principal's applicable sessions.

1462 4.4.4.2 <LogoutResponse> Usage

1463 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1464 entity; the Format attribute MUST be omitted or have a value of
1465 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1466 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1467 the message or using a binding-specific mechanism.

1468 4.4.5 Use of Metadata

1469 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported
1470 bindings and location(s) to which an entity may send requests and responses using this profile.

1471 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1472 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1473 settings to use, along with a public key to use in delivering a bulk encryption key.

1474 4.5 Name Identifier Management Profile

1475 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1476 some form of [E55]persistentlong-term identifier (including but not limited to identifiers with a Format of
1477 urn:oasis:names:tc:SAML:2.0:nameid-format:persistent) for a principal with a service
1478 provider, allowing them to share a common identifier for some length of time. Subsequently, the identity
1479 provider may wish to notify the service provider of a change in the [E12]format and/or value that it will use
1480 to identify the same principal in the future. Alternatively the service provider may wish to attach its own
1481 "alias" for the principal in order to ensure that the identity provider will include it when communicating with
1482 it in the future about the principal[E55]using that identifier. Finally, one of the providers may wish to inform
1483 the other that it will no longer issue or accept messages using a particular identifier. To implement these

scenarios, a profile of the SAML Name Identifier Management protocol is used.

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

4.5.1 Required Information

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

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

Description: Given below.

Updates: None.

4.5.2 Profile Overview

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

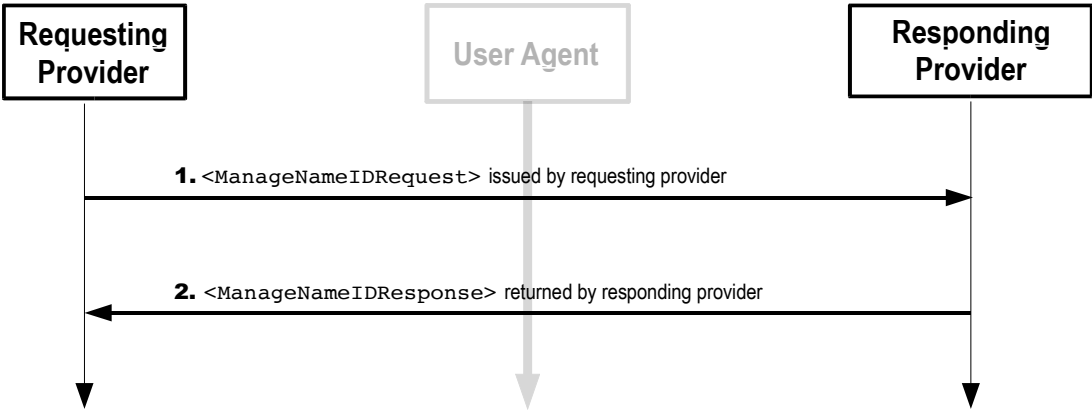


Figure 4

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

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

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

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

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

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

1511 the form of the request in step 1).

1512 4.5.3 Profile Description

1513 In the descriptions below, the following is referred to:

1514 Name Identifier Management Service

1515 This is the name identifier management protocol endpoint at an identity or service provider to which
1516 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
1517 representing them) are delivered. The same or different endpoints MAY be used for requests and
1518 responses.

1519 4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider

1520 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
1521 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1522 used to determine the location of this endpoint and the bindings supported by the responding provider.

1523 Synchronous Bindings (Back-Channel)

1524 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1525 send the request directly to the other provider. The requester MUST authenticate itself to the other
1526 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported
1527 mechanism.

1528 Asynchronous Bindings (Front-Channel)

1529 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1530 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1531 request to the other provider through the user agent.

1532 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1533 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1534 profile defined in Section 5 is used by the other provider, which makes a callback to the requesting
1535 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1536 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1537 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The
1538 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1539 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1540 request issuer when the artifact is dereferenced.

1541 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1542 associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1543 little information as possible in the RelayState value unless the use of the profile does not require
1544 such privacy measures.

1545 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section
1546 4.5.4.1.

1547 4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1548 Provider

1549 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1550 processing the message or upon encountering an error, the recipient MUST issue a
1551 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1552 to complete the SAML protocol exchange.

1553 Synchronous Bindings (Back-Channel)

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

Asynchronous Bindings (Front-Channel)

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

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

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

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

4.5.4 Use of Name Identifier Management Protocol

4.5.4.1 <ManageNameIDRequest> Usage

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

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

4.5.4.2 <ManageNameIDResponse> Usage

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

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

4.5.5 Use of Metadata

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

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

5 Artifact Resolution Profile

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

5.1 Required Information

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

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

Description: Given below.

Updates: None

5.2 Profile Overview

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

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

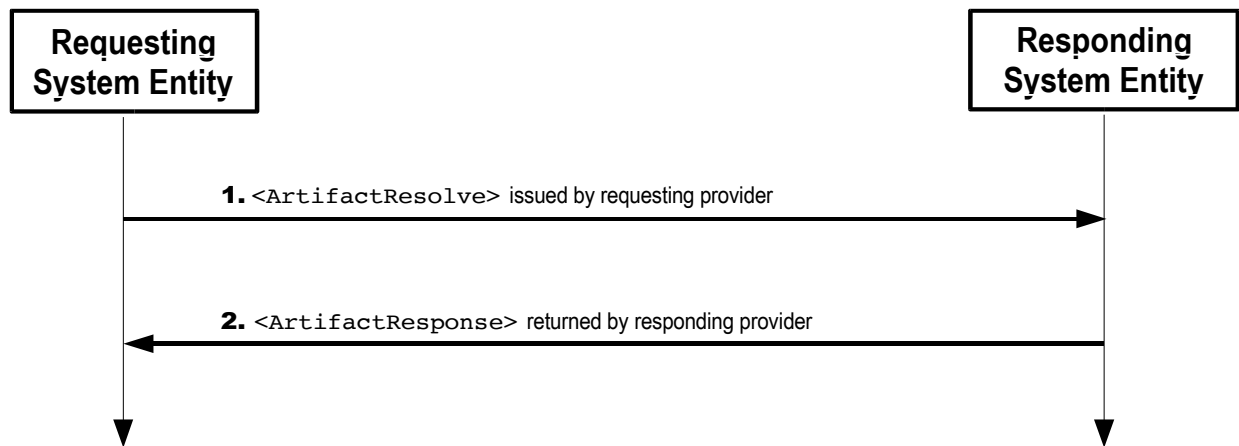


Figure 5

The following steps are described by the profile.

1. <ArtifactResolve> issued by Requesting Entity

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

2. <ArtifactResponse> issued by Responding Entity

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

5.3 Profile Description

In the descriptions below, the following is referred to:

Artifact Resolution Service

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

5.3.1 <ArtifactResolve> issued by Requesting Entity

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

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

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

5.3.2 <ArtifactResponse> issued by Responding Entity

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

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

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

5.4 Use of Artifact Resolution Protocol

5.4.1 <ArtifactResolve> Usage

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

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

5.4.2 <ArtifactResponse> Usage

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

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

5.5 Use of Metadata

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

6 Assertion Query/Request Profile

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

6.1 Required Information

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

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

Description: Given below.

Updates: None.

6.2 Profile Overview

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

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

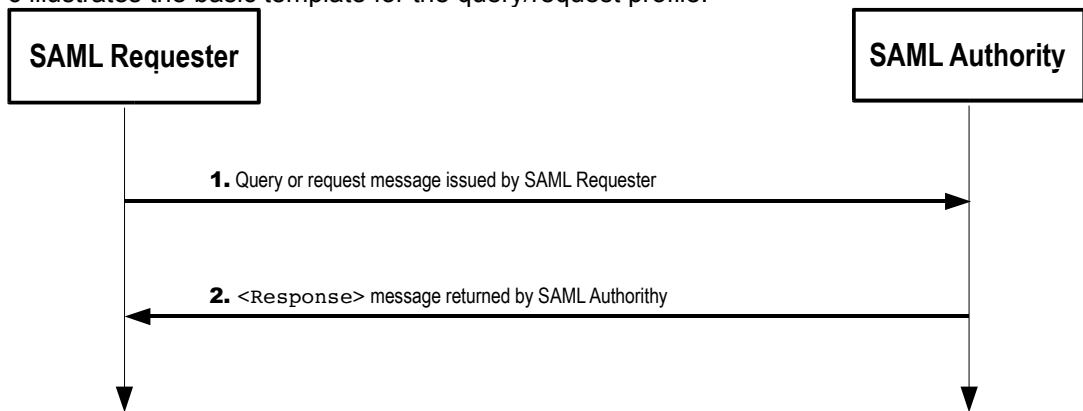


Figure 6

The following steps are described by the profile.

1. Query/Request issued by SAML Requester

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

2. <Response> issued by SAML Authority

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

6.3 Profile Description

In the descriptions below, the following are referred to:

Query/Request Service

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

6.3.1 Query/Request issued by SAML Requester

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

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

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

6.3.2 `<Response>` issued by SAML Authority

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

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

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

6.4 Use of Query/Request Protocol

6.4.1 Query/Request Usage

The `<Issuer>` element MUST be present.

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

6.4.2 `<Response>` Usage

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

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

[E93] Note that if `<EncryptedAssertion>` elements are present and a CBC-mode algorithm is used, then the `<Response>` SHOULD be signed to ensure the ciphertext is integrity protected (see section 6.2

1724 | of [SAMLCore]). Some deployments may require both the <Response> and any <Assertion> elements
1725 | be signed to address both the encryption issue and non-repudiation of the assertion (the latter being
1726 | outside the scope of SAML).

1727 | **6.5 Use of Metadata**

1728 | [SAMLMeta] defines several endpoint elements, <md:AssertionIDRequestService>,
1729 | <md:AuthnQueryService>, <md:AttributeService>, and <md:AuthzService>, to describe
1730 | supported bindings and location(s) to which a requester may send requests or queries using this profile.

1731 | The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1732 | use that entity's <md:KeyDescriptor> element with a use attribute of encryption to determine an
1733 | appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1734 | encryption key.

1735 | The various role descriptors MAY contain <md:NameIDFormat>, <md:AttributeProfile>, and
1736 | <saml:Attribute> elements (as applicable) to indicate the general ability to support particular name
1737 | identifier formats, attribute profiles, or specific attributes and values. The ability to support any such
1738 | features during a given request is dependent on policy and the discretion of the authority.

7 Name Identifier Mapping Profile

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

7.1 Required Information

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

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

Description: Given below.

Updates: None.

7.2 Profile Overview

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

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

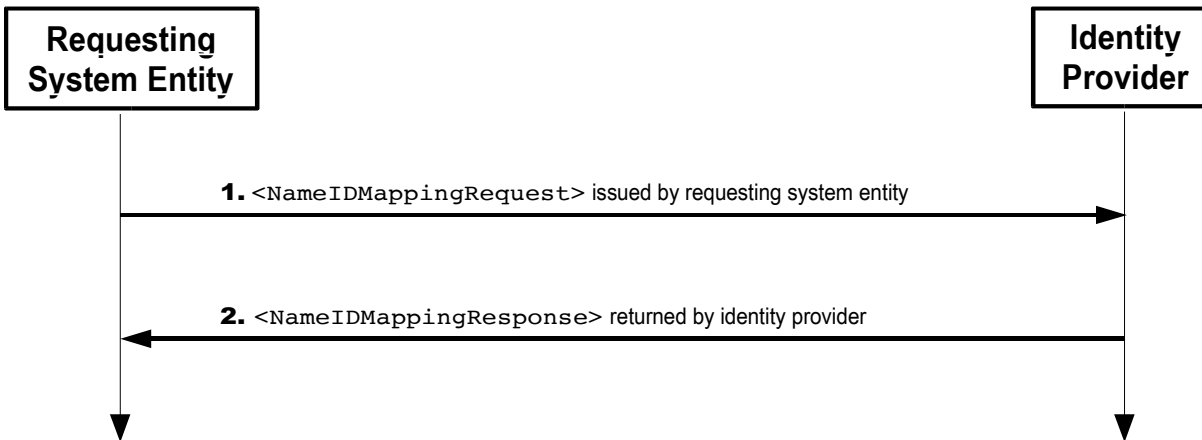


Figure 7

The following steps are described by the profile.

1. <NameIDMappingRequest> issued by Requesting Entity

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

2. <NameIDMappingResponse> issued by Identity Provider

In step 2, the responding identity provider (after processing the request) issues a

1761 <NameIDMappingResponse> message to the requester.

1762 **7.3 Profile Description**

1763 In the descriptions below, the following is referred to:

1764 **Name Identifier Mapping Service**

1765 This is the name identifier mapping protocol endpoint at an identity provider to which
1766 <NameIDMappingRequest> messages are delivered.

1767 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1768 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1769 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1770 location of this endpoint and the bindings supported by the identity provider.

1771 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1772 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1773 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1774 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1775 Section 7.4.1.

1776 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1777 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1778 After processing the message or upon encountering an error, the identity provider MUST return a
1779 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1780 complete the SAML protocol exchange.

1781 The responder MUST authenticate itself to the requester, either by signing the
1782 <NameIDMappingResponse> or using any other binding-supported mechanism.

1783 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1784 Section 7.4.2.

1785 **7.4 Use of Name Identifier Mapping Protocol**

1786 **7.4.1 <NameIDMappingRequest> Usage**

1787 The <Issuer> element MUST be present.

1788 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1789 the message or using a binding-specific mechanism.

1790 **7.4.2 <NameIDMappingResponse> Usage**

1791 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1792 identity provider; the `Format` attribute MUST be omitted or have a value of
1793 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1794 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1795 the message or using a binding-specific mechanism.

1796 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1797 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1798 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place
1799 it in subsequent protocol messages or assertions.

1800 **7.4.2.1 Limiting Use of Mapped Identifier**

1801 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1802 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1803 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1804 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1805 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1806 parties, and MUST be signed for integrity protection.

1807 **7.5 Use of Metadata**

1808 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1809 bindings and location(s) to which a requester may send requests using this profile.

1810 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1811 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1812 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

8 SAML Attribute Profiles

8.1 Basic Attribute Profile

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

8.1.1 Required Information

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

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

Description: Given below.

Updates: None.

8.1.2 SAML Attribute Naming

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

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

8.1.2.1 Attribute Name Comparison

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

8.1.3 Profile-Specific XML Attributes

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

8.1.4 SAML Attribute Values

The schema type of the contents of the <AttributeValue> element MUST be drawn from one of the types defined in Section 3[E51]-3 of [Schema2]. The xsi:type attribute MUST be present and be given the appropriate value.

8.1.5 Example

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

8.2 X.500/LDAP Attribute Profile [E53] – Deprecated

[E53]Note: This attribute profile is deprecated because of a flaw that makes it schema-invalid. The SSTC has replaced it with a separately published SAML V2.0 X.500/LDAP Attribute Profile specification that removes this flaw.

Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model information to be stored in these directories. In particular, in X.500, attribute type definitions are used to specify the syntax and other features of attributes, the basic information storage unit in a directory (this document refers to these as “directory attributes”). Directory attribute types are defined in schema in the X.500 and LDAP specifications themselves, schema in other public documents (such as the Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of these directory attribute types in the context of SAML attribute statements, without having to manually create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.

The X.500/LDAP attribute profile defines a common convention for the naming and representation of such attributes when expressed as SAML attributes.

8.2.1 Required Information

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

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

Description: Given below.

Updates: None.

8.2.2 SAML Attribute Naming

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

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

Example:

```
urn:oid:2.5.4.3
```

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

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

8.2.2.1 Attribute Name Comparison

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

8.2.3 Profile-Specific XML Attributes

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

8.2.4 SAML Attribute Values

Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax which specifies how attribute or assertion values conforming to the syntax are to be represented when transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to this profile may define attribute value formats for directory attributes whose syntaxes specify other encodings.

To represent the encoding rules in use for a particular attribute value, the `<AttributeValue>` element MUST contain an XML attribute named `Encoding` defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:x500`. (See [E53] for an issue with this attribute.)

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

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

Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
Bit String	1.3.6.1.4.1.1466.115.121.1.6
Boolean	1.3.6.1.4.1.1466.115.121.1.7
Country String	1.3.6.1.4.1.1466.115.121.1.11
DN	1.3.6.1.4.1.1466.115.121.1.12
Directory String	1.3.6.1.4.1.1466.115.121.1.15
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
IA5 String	1.3.6.1.4.1.1466.115.121.1.26
INTEGER	1.3.6.1.4.1.1466.115.121.1.27
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
Numeric String	1.3.6.1.4.1.1466.115.121.1.36
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
Octet String	1.3.6.1.4.1.1466.115.121.1.40
OID	1.3.6.1.4.1.1466.115.121.1.38
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
Postal Address	1.3.6.1.4.1.1466.115.121.1.41
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
Printable String	1.3.6.1.4.1.1466.115.121.1.44
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
UTC Time	1.3.6.1.4.1.1466.115.121.1.53

For all other LDAP syntaxes, the attribute value is encoded, as the content of the `<AttributeValue>` element, by base64-encoding [RFC2045] the [E48]encompassingcontents of the ASN.1 OCTET STRING-encoded LDAP attribute value (not including the ASN.1 OCTET STRING wrapper). The `xsi:type` XML attribute MUST be set to `xs:base64Binary`. The profile-specific `Encoding` XML attribute is provided, with a value of `"LDAP"`.

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

8.2.5 Profile-Specific Schema

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

```
<schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  <annotation>
    <documentation>
      Document identifier: saml-schema-x500-2.0
      Location: http://docs.oasis-open.org/security/saml/v2.0/
      Revision history:
        V2.0 (March, 2005):
          Custom schema for X.500 attribute profile, first published in
SAML 2.0.
    </documentation>
  </annotation>
  <attribute name="Encoding" type="string"/>
</schema>
```

8.2.6 Example

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

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

8.3 UUID Attribute Profile

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

8.3.1 Required Information

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

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

Description: Given below.

Updates: None.

8.3.2 UUID and GUID Background

UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to

1980 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs
1981 were originally used in the Network Computing System (NCS), and then used in the Open Software
1982 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1983 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1984 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1985 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1986 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1987 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1988 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1989 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

1990 **8.3.3 SAML Attribute Naming**

1991 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1992 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1993 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace
1994 described in [E70][RFC4122] is used. In this approach the `Name` XML
1995 attribute is based on the URN form of the underlying UUID that identifies the attribute.

1996 Example:

1997 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1998 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1999 in the `Name` XML attribute.

2000 For purposes of human readability, there may also be a requirement for some applications to carry an
2001 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
2002 [SAMLCore]) MAY be used for this purpose.

2003 **8.3.3.1 Attribute Name Comparison**

2004 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
2005 values are equal in the sense of <http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt> [E70]
2006 [RFC4122]. The `FriendlyName` attribute plays no role in the comparison.

2007 **8.3.4 Profile-Specific XML Attributes**

2008 No additional XML attributes are defined for use with the `<Attribute>` element.

2009 **8.3.5 SAML Attribute Values**

2010 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
2011 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST
2012 be set to `xs:anyURI`.

2013 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`
2014 element.

2015 **8.3.6 Example**

2016 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has
2017 a well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```

2018 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2019         Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"
2020         FriendlyName="pre_auth_req">
2021     <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>
2022 </saml:Attribute>

```

2023 8.4 DCE PAC Attribute Profile

2024 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
 2025 and values. It is used to standardize a mapping between the primary information that makes up a DCE
 2026 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
 2027 Section 8.3.

2028 8.4.1 Required Information

2029 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
 2030 assigned in the corresponding DCE PAC attribute profile schema document [SAML DCE-xsd])

2031 **Contact information:** security-services-comment@lists.oasis-open.org

2032 **Description:** Given below.

2033 **Updates:** None.

2034 8.4.2 PAC Description

2035 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
 2036 information is common across principals and makes up the bulk of a DCE identity:

- 2037 • The principal's DCE "realm" or "cell"
- 2038 • The principal's unique identifier
- 2039 • The principal's primary DCE local group membership
- 2040 • The principal's set of DCE local group memberships (multi-valued)
- 2041 • The principal's set of DCE foreign group memberships (multi-valued)

2042 The primary value(s) of each of these attributes is a UUID.

2043 8.4.3 SAML Attribute Naming

2044 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual
 2045 specific attribute names, rather than a naming convention.

2046 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST
 2047 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

2048 For purposes of human readability, there may also be a requirement for some applications to carry an
 2049 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
 2050 [SAML Core]) MAY be used for this purpose.

2051 See Section 8.4.6 for the specific attribute names defined by this profile.

2052 8.4.3.1 Attribute Name Comparison

2053 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute

values are equal in the sense of <http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt> [E70] [RFC4122]. The `FriendlyName` attribute plays no role in the comparison.

8.4.4 Profile-Specific XML Attributes

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

8.4.5 SAML Attribute Values

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

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

The following schema listing shows how the profile-specific XML attributes and complex type used in an `xsi:type` specification are defined [SAMLDCExsd]:

```
<schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
  xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  <annotation>
    <documentation>
      Document identifier: saml-schema-dce-2.0
      Location: http://docs.oasis-open.org/security/saml/v2.0/
      Revision history:
      V2.0 (March, 2005):
      Custom schema for DCE attribute profile, first published in
      SAML 2.0.
    </documentation>
  </annotation>
  <complexType name="DCEValueType">
    <simpleContent>
      <extension base="anyURI">
        <attribute ref="dce:Realm" use="optional"/>
        <attribute ref="dce:FriendlyName" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>
  <attribute name="Realm" type="anyURI"/>
  <attribute name="FriendlyName" type="string"/>
</schema>
```

8.4.6 Attribute Definitions

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

2102 Note that such use of `xsi:type` will require validating attribute consumers to include the extension
2103 schema defined by this profile.

2104 8.4.6.1 Realm

2105 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

2106 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

2107 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
2108 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
2109 realm's string name.

2110 8.4.6.2 Principal

2111 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

2112 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

2113 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
2114 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
2115 the principal's string name.

2116 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2117 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2118 8.4.6.1).

2119 8.4.6.3 Primary Group

2120 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

2121 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

2122 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
2123 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
2124 the group's string name.

2125 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2126 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2127 8.4.6.1).

2128 8.4.6.4 Groups

2129 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

2130 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

2131 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE group membership
2132 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
2133 the group's string name.

2134 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2135 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2136 8.4.6.1).

8.4.6.5 Foreign Groups

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

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

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

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

8.4.7 Example

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

```
<saml:Assertion xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
...>
  <saml:Issuer>...</saml:Issuer>
  <saml:Subject>...</saml:Subject>
  <saml:AttributeStatement>
    <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">
      <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="example.com">
        urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">
      <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">
        urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">
      <saml:AttributeValue xsi:type="dce:DCEValueType"
        dce:FriendlyName="cubicle-dwellers">
        urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">
      <saml:AttributeValue xsi:type="dce:DCEValueType"
        dce:FriendlyName="cubicle-dwellers">
        urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
      </saml:AttributeValue>
      <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="underpaid">
        urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-
groups">
      <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="engineers"
dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">
```

```

2191 urn:uuid:00099cfl-a355-10f9-9e95-004005b13a2b
2192 </saml:AttributeValue>
2193 </saml:Attribute>
2194 </saml:AttributeStatement>
2195 </saml:Assertion>

```

2196 8.5 XACML Attribute Profile

2197 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
 2198 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
 2199 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
 2200 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
 2201 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
 2202 XACML attributes and used as input to XACML authorization decisions.

2203 8.5.1 Required Information

2204 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace
 2205 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

2206 **Contact information:** security-services-comment@lists.oasis-open.org

2207 **Description:** Given below.

2208 **Updates:** None.

2209 8.5.2 SAML Attribute Naming

2210 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
 2211 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

2212 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

2213 For purposes of human readability, there may also be a requirement for some applications to carry an
 2214 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
 2215 [SAMLCore]) MAY be used for this purpose, but is not translatable into an XACML attribute equivalent.

2216 8.5.2.1 Attribute Name Comparison

2217 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
 2218 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

2219 8.5.3 Profile-Specific XML Attributes

2220 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
 2221 valued XML attribute called `DataType` is defined in the XML namespace
 2222 urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML.

2223 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified
 2224 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

2225 While in principle any URI reference can be used as a data type, the standard values to be used are
 2226 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
 2227 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
 2228 must be extended to support the new data types.

2229 8.5.4 SAML Attribute Values

2230 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed

2231 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
2232 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute
2233 SHOULD also be used on the `<AttributeValue>` element(s).

2234 8.5.5 Profile-Specific Schema

2235 The following schema listing shows how the profile-specific `DataType` XML attribute is defined
2236 [SAMLXAC-xsd]:

```
2237 <schema
2238   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2239   xmlns="http://www.w3.org/2001/XMLSchema"
2240   elementFormDefault="unqualified"
2241   attributeFormDefault="unqualified"
2242   blockDefault="substitution"
2243   version="2.0">
2244   <annotation>
2245     <documentation>
2246       Document identifier: saml-schema-xacml-2.0
2247       Location: http://docs.oasis-open.org/security/saml/v2.0/
2248       Revision history:
2249       V2.0 (March, 2005):
2250       Custom schema for XACML attribute profile, first published in
2251 SAML 2.0.
2252     </documentation>
2253   </annotation>
2254   <attribute name="DataType" type="anyURI"/>
2255 </schema>
```

2256 8.5.6 Example

2257 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
2258 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
2259 attribute profiles when they are compatible with each other.

```
2260 <saml:Attribute
2261   xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2262   xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2263   |   xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2264   |   [E39] ldapprof:Encoding="LDAP"
2265   |   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2266   |   Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2267   |   <saml:AttributeValue xsi:type="xs:string"
2268   |   |   ldapprof:Encoding="LDAP">By-Tor</saml:AttributeValue>
2269 </saml:Attribute>
```

9 References

- 2270
- 2271 **[AES]** FIPS-197, Advanced Encryption Standard (AES). See <http://www.nist.gov/>.
- 2272 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”.
2273 See <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2274 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic
2275 notation, ITU-T Recommendation X.680, July 2002. See
2276 [http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680)
2277 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2278 **[eduPerson]** eduPerson.Idif. See <http://www.educause.edu/eduperson>.
- 2279 **[LDAP]** J. Hodges et al. *Lightweight Directory Access Protocol (v3): Technical Specification*.
2280 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2281 ~~**[E70][Mealling]** P. Leach et al. *A UUID-URN Namespace*. IETF Internet Draft, December 2004. See~~
2282 ~~<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt>.~~
- 2283 **[MSURL]** Microsoft technical support article. See
2284 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2285 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See
2286 http://wp.netscape.com/newsref/std/cookie_spec.html.
- 2287 **[PAOS]** R. Aarts. *Liberty Reverse HTTP Binding for SOAP Specification* Version 1.0. Liberty
2288 Alliance Project, 2003. See <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>.
- 2289 **[Rescorla-Sec]** E. Rescorla et al. *Guidelines for Writing RFC Text on Security Considerations*. IETF
2290 RFC 3552, July 2003. See <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2291 **[RFC1738]** T. Berners-Lee et al. *Uniform Resource Locators (URL)*. IETF RFC 1738, December
2292 1994. See <http://www.ietf.org/rfc/rfc1738.txt>.
- 2293 **[RFC1750]** D. Eastlake et al. *Randomness Recommendations for Security*. IETF RFC 1750,
2294 December 1994. See <http://www.ietf.org/rfc/rfc1750.txt>.
- 2295 **[RFC1945]** T. Berners-Lee et al. *Hypertext Transfer Protocol – HTTP/1.0*. IETF RFC 1945, May
2296 1996. See <http://www.ietf.org/rfc/rfc1945.txt>.
- 2297 **[RFC2045]** N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of*
2298 *Internet Message Bodies*. IETF RFC 2045, November 1996. See
2299 <http://www.ietf.org/rfc/rfc2045.txt>.
- 2300 **[RFC2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC
2301 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 2302 **[RFC2246]** T. Dierks. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See
2303 <http://www.ietf.org/rfc/rfc2246.txt>.
- 2304 **[RFC2256]** M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*. IETF RFC
2305 2256, December 1997. See <http://www.ietf.org/rfc/rfc2256.txt>.
- 2306 **[RFC2279]** F. Yergeau. *UTF-8, a transformation format of ISO 10646*. IETF RFC 2279, January
2307 1998. See <http://www.ietf.org/rfc/rfc2279.txt>.
- 2308 **[RFC2616]** R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999.
2309 See <http://www.ietf.org/rfc/rfc2616.txt>.
- 2310 **[RFC2617]** J. Franks et al. *HTTP Authentication: Basic and Digest Access Authentication*. IETF
2311 RFC 2617, June 1999. See <http://www.ietf.org/rfc/rfc2617.txt>.
- 2312 **[RFC2798]** M. Smith. *Definition of the inetOrgPerson LDAP Object Class*. IETF RFC 2798, April
2313 2000. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2314 **[RFC2965]** D. Cristol et al. *HTTP State Management Mechanism*. IETF RFC 2965, October 2000.
2315 See <http://www.ietf.org/rfc/rfc2965.txt>.

2316	[RFC3061]	M. Mealling. <i>A URN Namespace of Object Identifiers</i> . IETF RFC 3061, February 2001. See http://www.ietf.org/rfc/rfc3061.txt .
2317		
2318	[E70][RFC4122]	P. Leach et al. <i>A Universally Unique Identifier (UUID) URN Namespace</i> . IETF RFC 4122, July 2005. See http://www.ietf.org/rfc/rfc4122.txt .
2319		
2320	[SAMLBind]	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/ .
2321		
2322		
2323	[SAMLConform]	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. See http://www.oasis-open.org/committees/security/ .
2324		
2325		
2326	[SAMLCore]	S. Cantor et al. <i>Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-core-2.0-os. See http://www.oasis-open.org/committees/security/ .
2327		
2328		
2329	[SAML DCE-xsd]	S. Cantor et al. SAML DCE PAC attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-dce-2.0. See http://www.oasis-open.org/committees/security/ .
2330		
2331		
2332	[SAML ECP-xsd]	S. Cantor et al. SAML ECP profile schema. OASIS SSTC, March 2005. Document ID saml-schema-ecp-2.0. See http://www.oasis-open.org/committees/security/ .
2333		
2334	[SAML Gloss]	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/ .
2335		
2336		
2337	[SAML X500-xsd]	S. Cantor et al. SAML X.500/LDAP attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-x500-2.0. See http://www.oasis-open.org/committees/security/ .
2338		
2339		
2340	[SAML Meta]	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/ .
2341		
2342		
2343	[SAML Reqs]	Darren Platt et al. <i>OASIS Security Services Use Cases and Requirements</i> . OASIS SSTC, May 2001. Document ID draft-sstc-saml-reqs-01. See http://www.oasis-open.org/committees/security/ .
2344		
2345		
2346	[SAML Sec]	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/ .
2347		
2348		
2349	[SAML Web]	OASIS Security Services Technical Committee website, http://www.oasis-open.org/committees/security/ .
2350		
2351	[SAML XAC-xsd]	S. Cantor et al. SAML XACML attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-xacml-2.0. See http://www.oasis-open.org/committees/security/ .
2352		
2353		
2354	[Schema1]	H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/ . Note that this specification normatively references [Schema2], listed below.
2355		
2356		
2357	[Schema2]	Paul V. Biron, Ashok Malhotra. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-2/ .
2358		
2359	[SESSION]	RL 'Bob' Morgan. <i>Support of target web server sessions in Shibboleth</i> . Shibboleth, May 2001. See http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt .
2360		
2361		
2362	[ShibMarlena]	Marlena Erdos et al. <i>Shibboleth Architecture DRAFT v05</i> . Shibboleth, May 2002. See http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html .
2363		
2364	[SOAP1.1]	D. Box et al. <i>Simple Object Access Protocol (SOAP) 1.1</i> . World Wide Web Consortium Note, May 2000. See http://www.w3.org/TR/SOAP .
2365		
2366	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.

2367	[WEBSSO]	RL 'Bob' Morgan. <i>Interactions between Shibboleth and local-site web sign-on services</i> . Shibboleth, April 2001. See http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt .
2368		
2369		
2370	[X.500]	Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services. ITU-T Recommendation X.500, February 2001. See http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.500 .
2371		
2372		
2373		
2374	[XMLEnc]	D. Eastlake et al. <i>XML Encryption Syntax and Processing</i> . World Wide Web Consortium Recommendation, December 2002. See http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/ .
2375		
2376		
2377	[XMLSig]	D. Eastlake et al. <i>XML-Signature Syntax and Processing</i> . [E74]Second Edition . World Wide Web Consortium Recommendation, June 2008 February 2002. See http://www.w3.org/TR/xmldsig-core/ .
2378		
2379		
2380	[XACML]	T. Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Versions 1.0, 1.1, and 2.0. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml .
2381		
2382		

Appendix A. Acknowledgments

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

- Conor Cahill, AOL
- John Hughes, Atos Origin
- Hal Lockhart, BEA Systems
- Mike Beach, Boeing
- Rebekah Metz, Booz Allen Hamilton
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Thomas Wisniewski, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Guy Denton, IBM
- Heather Hinton, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Peter Davis, Neustar
- Jeff Hodges, Neustar
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Abbie Barbir, Nortel Networks
- Scott Kiestler, Novell
- Cameron Morris, Novell
- Paul Madsen, NTT
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Prateek Mishra, Principal Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Eve Maler, Sun Microsystems

- 2426 • Ronald Monzillo, Sun Microsystems
- 2427 • Emily Xu, Sun Microsystems
- 2428 • Greg Whitehead, Trustgenix

2429 The editors also would like to acknowledge the following former SSTC members for their contributions to
 2430 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 2431 • Stephen Farrell, Baltimore Technologies
- 2432 • David Orchard, BEA Systems
- 2433 • Krishna Sankar, Cisco Systems
- 2434 • Zahid Ahmed, CommerceOne
- 2435 • Tim Alsop, CyberSafe Limited
- 2436 • Carlisle Adams, Entrust
- 2437 • Tim Moses, Entrust
- 2438 • Nigel Edwards, Hewlett-Packard
- 2439 • Joe Pato, Hewlett-Packard
- 2440 • Bob Blakley, IBM
- 2441 • Marlena Erdos, IBM
- 2442 • Marc Chanliau, Netegrity
- 2443 • Chris McLaren, Netegrity
- 2444 • Lynne Rosenthal, NIST
- 2445 • Mark Skall, NIST
- 2446 • Charles Knouse, Oblix
- 2447 • Simon Godik, Overxeer
- 2448 • Charles Norwood, SAIC
- 2449 • Evan Prodromou, Securant
- 2450 • Robert Griffin, RSA Security (former editor)
- 2451 • Sai Allarvarpu, Sun Microsystems
- 2452 • Gary Ellison, Sun Microsystems
- 2453 • Chris Ferris, Sun Microsystems
- 2454 • Mike Myers, Traceroute Security
- 2455 • Phillip Hallam-Baker, VeriSign (former editor)
- 2456 • James Vanderbeek, Vodafone
- 2457 • Mark O'Neill, Vordel
- 2458 • Tony Palmer, Vordel

2459 Finally, the editors wish to acknowledge the following people for their contributions of material used as
 2460 input to the OASIS Security Assertions Markup Language specifications:

- 2461 • Thomas Gross, IBM
- 2462 • Birgit Pfitzmann, IBM

2463 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on
 2464 the SSTC was the primary editor of the errata working document and who made major substantive
 2465 contributions to all of the errata materials.

Appendix B. Notices

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

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

Copyright © OASIS Open 2005. All Rights Reserved.

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

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

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