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

Committee Draft 02, 24 September 2004

Document identifier:
sstc-saml-profiles-2.0-cd-02

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

Editors:
John Hughes, Atos Origin
Scott Cantor, Internet2
Prateek Mishra, Netegrity
Frederick Hirsch, Nokia
Rob Philpott, RSA Security
Jeff Hodges, Sun Microsystems
Eve Maler, Sun Microsystems

SAML V2.0 Contributors:
Conor P. Cahill, AOL
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rick Randall, Booze, Allen, Hamilton
Tim Alsop, CyberSafe Limited
Nick Ragouzis, Enosis
John Hughes, Atos Origin
Paul Madsen, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Rebekah Metz, NASA
Prateek Mishra, Netegrity
Peter C Davis, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems

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

Abstract:

49

This 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.

50
51
52

Status:

53

This is a **second Committee Draft** approved by the Security Services Technical Committee on 21 September 2004.

54
55

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

56
57
58
59
60

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

61
62
63
64

Table of Contents

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.

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

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 described in IETF RFC 2119 [RFC2119].

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

```
Example code listings appear like this.
```

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

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

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

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

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.

4. Rules for determining the equality of SAML `<Attribute>` elements as defined by the profile, for

310   use when processing attributes, queries, etc.

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

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 corresponds to 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 `<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 a specified key is considered to be the 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.

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>
```

3.2 Sender Vouches

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

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

3.3 Bearer

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

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

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

```
<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <SubjectConfirmationData InResponseTo="_1234567890"
                Recipient="https://www.serviceprovider.com/saml/consumer"
                NotOnOrAfter="2004-03-19T13:27:00Z">
        </SubjectConfirmationData>
</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.
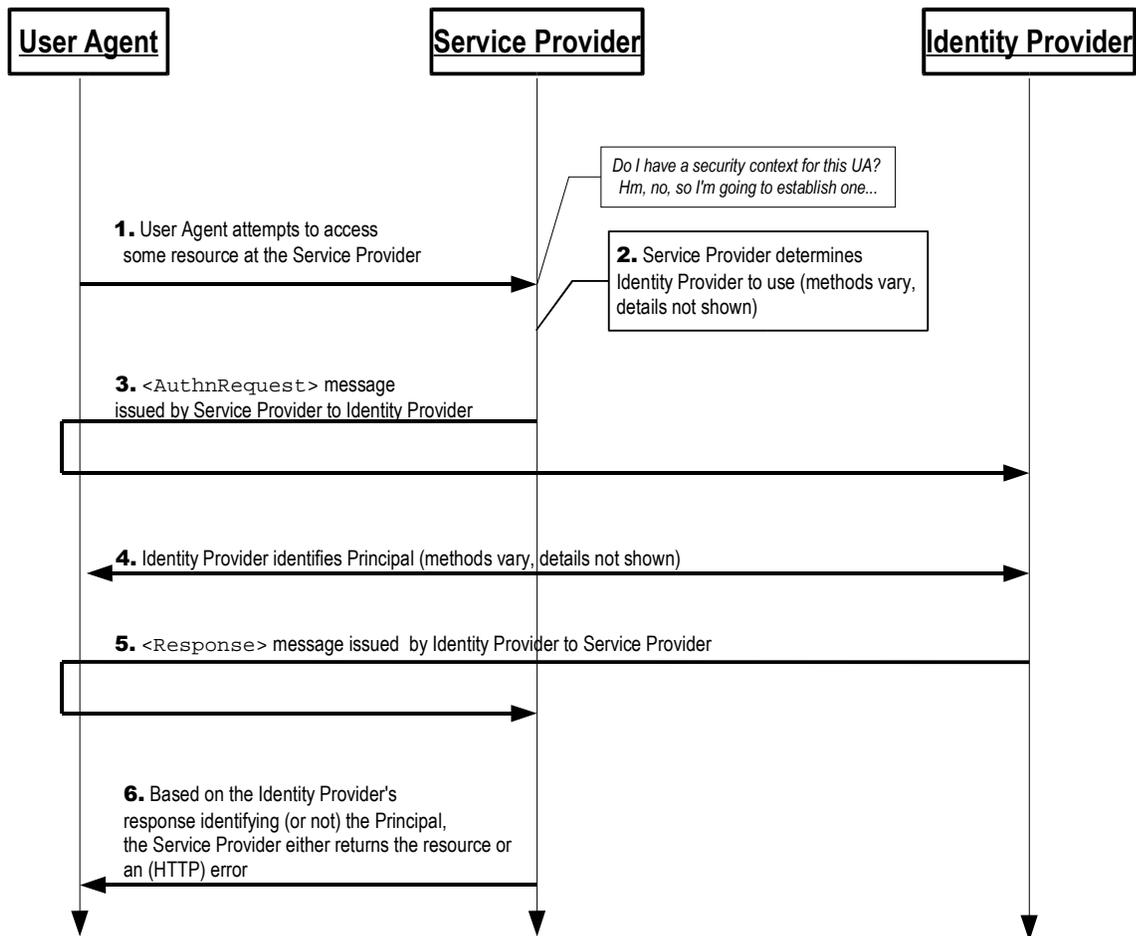
Figure 1

1. HTTP Request to Service Provider

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

2. Service Provider Determines Identity Provider

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

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

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

4. Identity Provider identifies Principal

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

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

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

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

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

4.1.3 Profile Description

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

433 **Single Sign-On Service**

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

436 **Assertion Consumer Service**

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

4.1.3.1 HTTP Request to Service Provider

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

4.1.3.2 Service Provider Determines Identity Provider

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

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

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

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

466 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or TLS
467 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The `<AuthnRequest>` message MAY
468 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
469 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

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

4.1.3.4 Identity Provider Identifies Principal

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

4.1.3.5 Identity Provider Issues <Response> to Service Provider

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

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

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

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

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

4.1.3.6 Service Provider Grants or Denies Access to User Agent

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

4.1.4 Use of Authentication Request Protocol

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

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.

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 insure 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:

- The `<Issuer>` element MAY be omitted, but 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 issuing identity provider; the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format: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.

- Any bearer `<SubjectConfirmationData>` elements 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 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`.

- Other statements and confirmation methods MAY be included in the 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.

- The assertion(s) containing a bearer subject confirmation MUST contain an `<AudienceRestriction>` including the service provider's unique identifier as an `<Audience>`.

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

4.1.4.3 <Response> Message Processing Rules

567 Regardless of the SAML binding used, the service provider MUST do the following:
568 • Verify any signatures present on the assertion(s) or the response

569 • Verify that the `Recipient` attribute in any bearer `<SubjectConfirmationData>` matches the
570   assertion consumer service URL to which the `<Response>` or artifact was delivered

571 • Verify that the `NotOnOrAfter` attribute in any bearer `<SubjectConfirmationData>` has not
572   passed, subject to allowable clock skew between the providers

573 • Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the `ID`
574   of its original `<AuthnRequest>` message, unless the response is unsolicited (see Section 4.5) in
575   which case the attribute MUST NOT be present

576 • Verify that any assertions relied upon are valid in other respects

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

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

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

4.1.4.4 Artifact-Specific <Response> Message Processing Rules

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

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

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

4.1.4.5 POST-Specific Processing Rules

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

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

4.1.5 Unsolicited Responses

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

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

603 of the service provider designated as the default.

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

607 4.1.6 Use of Metadata

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

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

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

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

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

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

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

635 4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

648   **Note:** The means by which a p[rincipal authenticates with an identity provider is outside of the
649   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 witih 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 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.

Figure 2 shows the processing flow in the ECP profile.

```
  Enhanced Client          Service Provider        Identity Provider
  or Proxy (ECP)
```

1. ECP attempts access of some resource
at the Service Provider via an HTTP request
with HTTP headers denoting the request as
being from an ECP

*Do I have a security
context for this ECP and
Principal?
Hm, no, so I'm going to
establish one.*

2. `<AuthnRequest>` message
issued by Service Provider using PAOS binding (i.e.
HTTP 200 OK response)

3. ECP determines
Identity Provider to
use (methods vary,
details not shown)

4. ECP issues `<AuthnRequest>` to the identity
provider using SAML SOAP binding

5. Identity Provider identifies Principal (methods vary, details not shown)

6. Identity Provider issues `<Response>` message to ECP using SAML SOAP
binding, targeted at service provider

7. ECP conveys `<Response>` message to
service provider using PAOS binding (i.e.. HTTP
POST)

8. Based on the Identity Provider's
response identifying (or not) the Principal,
the Service Provider either returns the resource,
or an (HTTP) error, in an HTTP response (e.g.
HTTP 200 OK)

Figure 2

Figure 2 illustrates the basic template for SSO using an ECP. 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. ECP issues HTTP Request to Service Provider

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

2. Service Provider issues `<AuthnRequest>` to ECP

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

3. ECP Determines Identity Provider

In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication
request protocol that supports its preferred binding. The means by which this is accomplished is

709 implementation-dependent. The ECP MAY use the SAML identity provider discovery profile
710 described in Section 4.3.

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

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

714 **5. Identity Provider identifies Principal**

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

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

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

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

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

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

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

4.2.3 Profile Description

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

4.2.3.1 ECP issues HTTP Request to Service Provider

732 The ECP sends an HTTP request to a service provider, specifying some resource. This HTTP request
733 MUST conform to the PAOS binding, which means it must include the following HTTP header fields:
734   1. The HTTP `Accept` Header field indicating the ability to accept the MIME type
735   "`application/vnd.paos+xml`"

736   2. The HTTP `PAOS` Header field specifying the PAOS version with urn:liberty:paos:2003-08 at
737   minimum.

738   3. Furthermore, support for this profile MUST be specified in the HTTP `PAOS` Header field as a service
739   value, with the value `urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp`. This value should
740   correspond to the service attribute in the PAOS Request SOAP header block

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

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

4.2.3.2 Service Provider Issues <AuthnRequest> to ECP

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

752 The SOAP envelope MUST contain:
753   1. An `<AuthnRequest>` element in the SOAP body, intended for the ultimate SOAP recipient, the

754   identity provider.

2. A PAOS SOAP header block targeted at the ECP using the SOAP `actor` value of `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control information such as the URL to which to send the response in this solicit-response message exchange pattern.

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

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

4.2.3.3 ECP Determines Identity Provider

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

4.2.3.4 ECP issues <AuthnRequest> to Identity Provider

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

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

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

4.2.3.5 Identity Provider Identifies Principal

At any time during the previous step or subequent 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.

4.2.3.6 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.

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

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

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

4.2.3.8 Service Provider Grants or Denies Access to Principal

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

4.2.4 ECP Profile Schema Usage

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

```
820  <schema
821        targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
822        xmlns="http://www.w3.org/2001/XMLSchema"
823        xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
824        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
825        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
826        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
827        elementFormDefault="unqualified"
828        attributeFormDefault="unqualified"
829        blockDefault="substitution"
830        version="2.0">
831        <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
832              schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
833        <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
834              schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
835        <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
836              schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>

837        <element name="Request" type="ecp:RequestType"/>
838        <complexType name="RequestType">
839              <sequence>
840                    <element ref="saml:Issuer"/>
841                    <element ref="samlp:IDPList" minOccurs="0"/>
842  </sequence>
843              <attribute ref="SOAP-ENV:mustUnderstand" use="required"/>
844              <attribute ref="SOAP-ENV:actor" use="required"/>
845              <attribute name="ProviderName" type="string" use="optional"/>
846              <attribute name="IsPassive" type="boolean" use="optional"/>
847        </complexType>

848        <element name="Response" type="ecp:ResponseType"/>
849        <complexType name="ResponseType">
850              <attribute ref="SOAP-ENV:mustUnderstand" use="required"/>
```

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

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

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

4.2.4.1 PAOS Request Header Block: SP to ECP

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

870 `responseConsumerURL` [Required]

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

876 `service` [Required]

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

879 `SOAP-ENV:mustUnderstand` [Required]

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

882 `SOAP-ENV:actor` [Required]

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

884 `messageID` [Optional]

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

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

4.2.4.2 ECP Request Header Block : SP to ECP

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

893 `SOAP-ENV:mustUnderstand` [Required]

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

896 `SOAP-ENV:actor` [Required]

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

898   `ProviderName` [Optional]

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

900   `IsPassive` [Optional]

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

904   `<saml:Issuer>` [Required]

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

908   `<samlp:IDPList>` [Optional]

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

911   ## 4.2.4.3 ECP RelayState Header Block: SP to ECP

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

915   `SOAP-ENV:mustUnderstand` [Required]

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

917   `SOAP-ENV:actor` [Required]

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

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

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

```
925    <SOAP-ENV:Envelope
926          xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
927          xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
928          xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
929      <SOAP-ENV:Header>
930        <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
931          responseConsumerURL="http://identity-service.example.com/abc"
932          messageID="6c3a4f8b9c2d" SOAP-
933    ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
934    ENV:mustUnderstand="1"
935          service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
936        </paos:Request>
937        <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
938          SOAP-ENV:mustUnderstand="1" SOAP-
939    ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
940          ProviderName="Service Provider X" IsPassive="0">
941          <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
942          <samlp:IDPList>
943            <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
944                Name="Identity Provider X"
945                Loc="https://IdentityProvider.example.com/saml2/sso"
```

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

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

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

4.2.4.4 ECP Response Header Block : IdP to ECP

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

974 `SOAP-ENV:mustUnderstand` [Required]

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

977 `SOAP-ENV:actor` [Required]

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

979 `AssertionConsumerServiceURL` [Required]

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

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

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

989 The ECP Response SOAP header has no element content.

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

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

```
996       <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
997  ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
998  AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_
999  consumer"/>
1000   </SOAP-ENV:Header>
1001   <SOAP-ENV:Body>
1002     <samlp:Response> ... </samlp:Response>
1003   </SOAP-ENV:Body>
1004  </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]

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

`SOAP-ENV:actor` [Required]

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

`refToMessageID` [Optional]

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

The PAOS Response SOAP header has no element content.

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

```
1019  <SOAP-ENV:Envelope
1020        xmlns:paos="urn:liberty:paos:2003-08"
1021        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1022        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1023   <SOAP-ENV:Header>
1024     <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1025  ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1026  ENV:mustUnderstand="1"/>
1027     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1028        SOAP-ENV:mustUnderstand="1" SOAP-
1029  ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1030        ...
1031     </ecp:RelayState>
1032   </SOAP-ENV:Header>
1033   <SOAP-ENV:Body>
1034     <samlp:Response> ... </samlp:Response>
1035   </SOAP-ENV:Body>
1036  </SOAP-ENV:Envelope>
```

4.2.5 Security Considerations

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

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

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

1046 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
1047 session.

4.3 Identity Provider Discovery Profile

1049 This section defines a profile by which a service provider can discover which identity providers a principal
1050 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
1051 service providers need a means to discover which identity provider(s) a principal uses. The discovery
1052 profile relies on a cookie that is written in a domain that is common between identity providers and service
1053 providers in a deployment. The domain that the deployment predetermines is known as the common
1054 domain in this profile, and the cookie containing the list of identity providers is known as the common
1055 domain cookie.

1056 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
1057 this profile.

4.3.1 Common Domain Cookie

1059 The name of the cookie MUST be _saml_idp. The format of the cookie value MUST be a set of one or
1060 more base-64 encoded URI values separated by a single space character. Each URI is the unique
1061 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
1062 URL encoded.

1063 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
1064 identifier to the list. If the identifier is already present in the list, it MAY remove and append it when
1065 authentication of the principal occurs. The intent is that the most recently established identity provider
1066 session is the last one in the list.

1067 The cookie MUST be set with no Path prefix or a Path prefix of "/". The Domain MUST be set to
1068 "[common-domain]" where [common-domain] is the common domain established within the deployment
1069 for use with this profile. The cookie MUST be marked as secure.

1070 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY
1071 be either session-only or persistent. This choice may be made within a deployment, but should apply
1072 uniformly to all identity providers in the deployment.

4.3.2 Setting the Common Domain Cookie

1074 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
1075 which the identity provider sets the cookie are implementation-specific so long as the cookie is
1076 successfully set with the parameters given above. One possible implementation strategy follows and
1077 should be considered non-normative. The identity provider may:
1078 • Have previously established a DNS and IP alias for itself in the common domain.

1079 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1080   scheme. The structure of the URL is private to the implementation and may include session
1081   information needed to identify the user-agent.

1082 • Set the cookie on the redirected user agent using the parameters specified above.

1083 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

4.3.3 Obtaining the Common Domain Cookie

1085 When a service provider needs to discover which identity providers a principal uses, it invokes an
1086 exchange designed to present the common domain cookie to the service provider after it is read by an
1087 HTTP server in the common domain.

1088 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
1089 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
1090 `<AuthnRequest>` to the identity provider for an optimized single sign-on process.

1091 The specific means by which the service provider reads the cookie are implementation-specific so long as
1092 it is able to cause the user agent to present cookies that have been set with the parameters given in
1093 Section 4.3.1. One possible implementation strategy is described as follows and should be considered
1094 non-normative. Additionally, it may be sub-optimal for some applications.
1095 • Have previously established a DNS and IP alias for itself in the common domain.

1096 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1097   scheme. The structure of the URL is private to the implementation and may include session
1098   information needed to identify the user-agent.

1099 • Set the cookie on the redirected user agent using the parameters specified above.

1100 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1101 4.4 Single Logout Profile

1102 Once a principal has authenticated to an identity provider, the authenticating entity may establish a
1103 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-
1104 specific means). The identity provider may subsequently issue assertions to service providers or other
1105 relying parties, based on this authentication event; a relying party may use this to establish *its own* session
1106 with the user.

1107 In such a situation, the identity provider can act as a session authority and the relying parties as session
1108 participants. At some later time, the principal may wish to terminate his or her session either with an
1109 individual session participant, or with all session participants in a given session managed by the session
1110 authority. The former case is considered out of scope of this specification. The latter case, however, may
1111 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1112 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this
1113 "global" session either by contacting the session authority, or an individual session participant. Also note
1114 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1115 which it is the relying party for another identity provider's assertions regarding that principal.

1116 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1117 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1118 front-channel binding may be required, for example, in cases in which a principal's session state exists
1119 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1120 session participant or session authority is required.

1121 4.4.1 Required Information

1122 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1123 **Contact information:** security-services-comment@lists.oasis-open.org

1124 **Description:** Given below.

1125 **Updates:** None

1126 4.4.2 Profile Overview

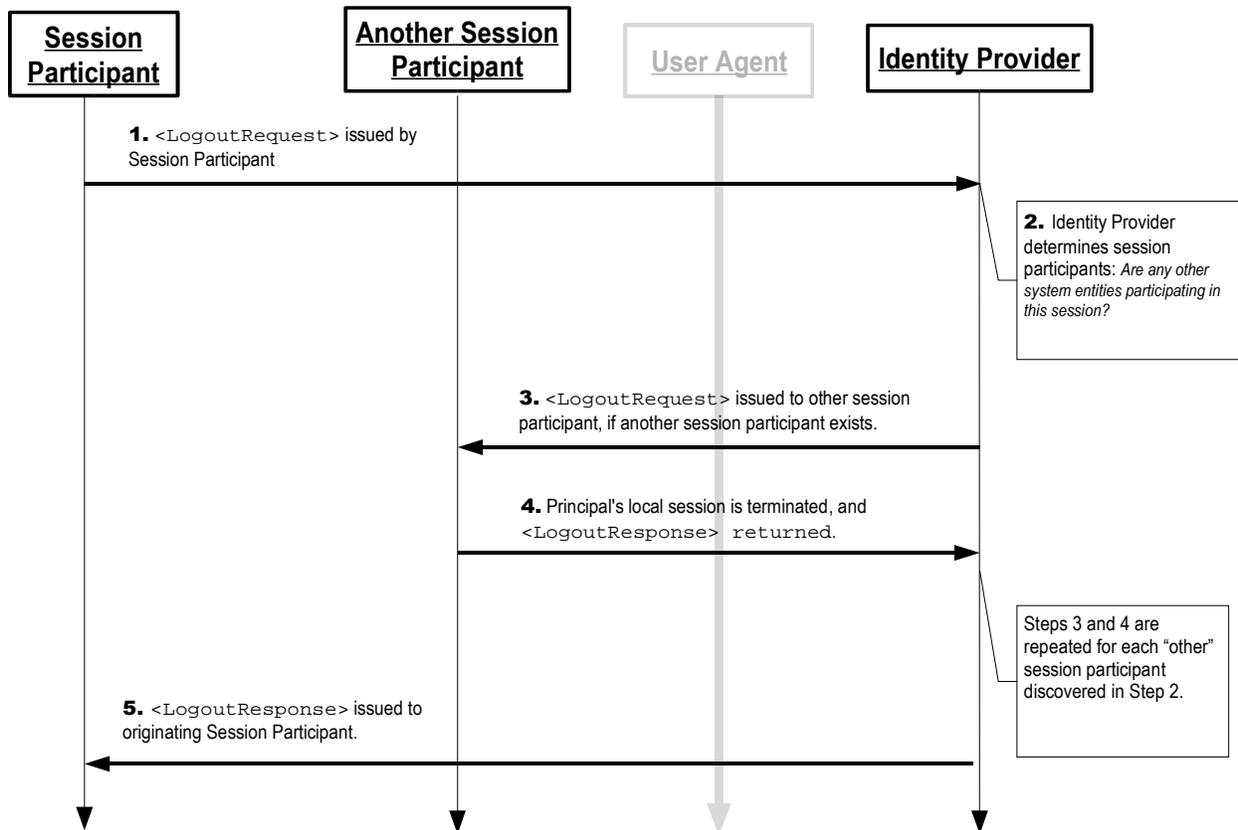1127 Figure 3 illustrates the basic template for achieving single logout:

Figure 3

1128 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
1129 may be a direct exchange between system entities, depending on the SAML binding used to implement
1130 the profile.

1131 The following steps are described by the profile. Within an individual step, there may be one or more
1132 actual message exchanges depending on the binding used for that step and other implementation-
1133 dependent behavior.

1134 **1. \<LogoutRequest> issued by Session Participant to Identity Provider**

1135   In step 1, the session participant initiates single logout and terminates a principal's session(s) by
1136   sending a `<LogoutRequest>` message to the identity provider from whom it received the
1137   corresponding authentication assertion. The request may be sent directly to the identity provider
1138   or sent indirectly through the user agent.

1139 **2. Identity Provider determines Session Participants**

1140   In step 2, the identity provider uses the contents of the `<LogoutRequest>` message (or if
1141   initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
1142   there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
1143   are repeated for each session participant identified.

1144 **3. \<LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1145   In step 3, the identity provider issues a `<LogoutRequest>` message to a session participant or
1146   session authority related to one or more of the session(s) being terminated. The request may be
1147   sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
1148   request in step 1).

1149 **4. Session Participant/Authority issues \<LogoutResponse> to Identity Provider**

1150   In step 4, a session participant or session authority terminates the principal's session(s) as
1151   directed by the request (if possible) and returns a `<LogoutResponse>` to the identity provider.

1152 The response may be returned directly to the identity provider or indirectly through the user agent
1153 (if consistent with the form of the request in step 3).

1154 **5. Identity Provider issues <LogoutResponse> to Session Participant**

1155 In step 5, the identity provider issues a `<LogoutResponse>` message to the original requesting
1156 session participant. The response may be returned directly to the session participant or indirectly
1157 through the user agent (if consistent with the form of the request in step 1).

1158 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1159 `<LogoutRequest>` to all session participants, also skipping step 5.

4.4.3 Profile Description

1161 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1162 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1163 **Single Logout Service**

1164 This is the single logout protocol endpoint at an identity provider or session participant to which the
1165 `<LogoutRequest>` or `<LogoutResponse>` messages (or an artifact representing them) are
1166 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

1168 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it
1169 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it
1170 received from the identity provider. If multiple identity providers are involved, then the profile MUST be
1171 repeated independently for each one.

1172 To initiate the profile, the session participant issues a `<LogoutRequest>` message to the identity
1173 provider's single logout service request endpoint containing one or more applicable `<SessionIndex>`
1174 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to
1175 determine the location of this endpoint and the bindings supported by the identity provider.

1176 **Synchronous Bindings (Back-Channel)**

1177 The session participant MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1178 send the request directly to the identity provider. The identity provider would then propagate any
1179 required logout messages to additional session participants as required using a synchronous binding.
1180 The requester MUST authenticate itself to the identity provider, either by signing the
1181 `<LogoutRequest>` or using any other binding-supported mechanism.

1182 **Asynchronous Bindings (Front-Channel)**

1183 Alternatively, the session participant MAY (if the principal's user agent is present) use an
1184 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1185 request to the identity provider through the user agent.

1186 If the HTTP Redirect or POST binding is used, then the `<LogoutRequest>` message is delivered to
1187 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1188 defined in Section 5 is used by the identity provider, which makes a callback to the session participant
1189 to retrieve the `<LogoutRequest>` message, using for example the SOAP binding.

1190 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1191 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The `<LogoutRequest>`
1192 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1193 if used, also provides for an alternate means of authenticating the request issuer when the artifact is
1194 dereferenced.

1195 Each of these bindings provide a RelayState mechanism that the session participant MAY use to
1196 associate the profile exchange with the original request. The session participant SHOULD reveal as
1197 little information as possible in the RelayState value unless the use of the profile does not require such
1198 privacy measures.

1199 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

1201 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1202 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1203 principal identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1204 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1205 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2
1206 of [SAMLCore].

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

1209 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1210 participant in a session being terminated. The request is sent in the same fashion as described in step 1
1211 using a SAML binding consistent with the capability of the responder and the availability of the user agent
1212 at the identity provider.

1213 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

1216 The session participant/authority MUST process the <LogoutRequest> message as defined in
1217 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1218 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1219 to complete the SAML protocol exchange.

Synchronous Bindings (Back-Channel)

1221 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1222 response is returned directly to complete the synchronous communication. The responder MUST
1223 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1224 using any other binding-supported mechanism.

Asynchronous Bindings (Front-Channel)

1226 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact
1227 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to
1228 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be
1229 used to determine the location of this endpoint and the bindings supported by the identity provider.

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

1234 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1235 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutResponse>
1236 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1237 if used, also provides for an alternate means of authenticating the response issuer when the artifact is
1238 dereferenced.

1239 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1240 4.4.4.2.

4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant

After processing the original session participant's `<LogoutRequest>` in step 1, or upon encountering an error, 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 in the same fashion as described in step 4, using a SAML binding consistent with the binding used in the request, the capability of the responder, and the availability of the user agent at the identity provider.

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

4.4.4 Use of Single Logout Protocol

4.4.4.1 <LogoutRequest> 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.

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

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

4.4.4.2 <LogoutResponse> 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.

4.4.5 Use of Metadata

[SAMLMeta] defines an endpoint element, `<md:SingleLogoutService>`, 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.

4.5 Name Identifier Management Profile

In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged some form of persistent identifier for a principal with a service provider, allowing them to share a common identifier for some length of time. Subsequently, the identity provider may wish to notify the service provider of a change in the format and/or value that it will use to identify the same principal in the future. Alternatively the service provider may wish to attach its own "alias" for the principal in order to insure that the identity provider will include it when communicating with it in the future about the principal. Finally, one of the providers may wish to inform 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

1284 protocol is used.

1285 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1286 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1287 front-channel binding may be required, for example, in cases in which direct interaction between the user
1288 agent and the responding provider is required in order to effect the change.

4.5.1 Required Information

1290 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1291 **Contact information:** security-services-comment@lists.oasis-open.org

1292 **Description:** Given below.

1293 **Updates:** None.

4.5.2 Profile Overview

1295 Figure 4 illustrates the basic template for the name identifier management profile.
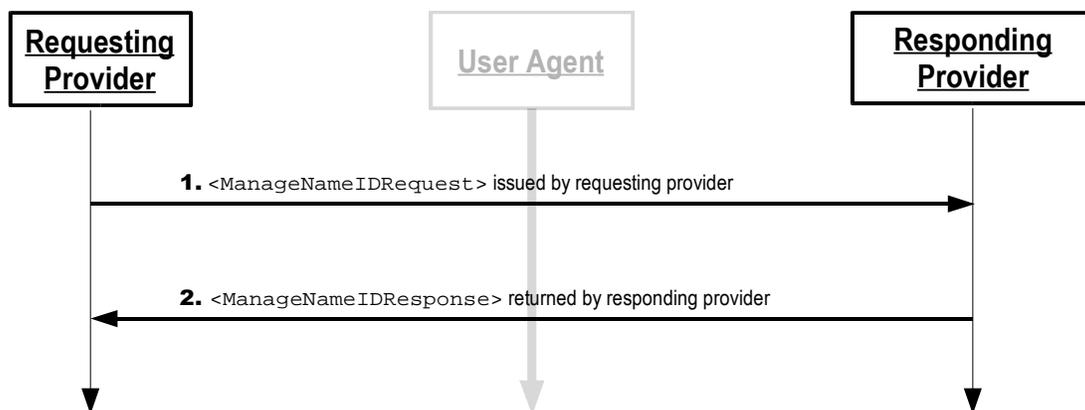

Figure 4

1296 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
1297 may be a direct exchange between system entities, depending on the SAML binding used to implement
1298 the profile.

1299 The following steps are described by the profile. Within an individual step, there may be one or more
1300 actual message exchanges depending on the binding used for that step and other implementation-
1301 dependent behavior.

1302 **1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1303 In step 1, an identity or service provider initiates the profile by sending a
1304 `<ManageNameIDRequest>` message to another provider that it wishes to inform of a change.
1305 The request may be sent directly to the responding provider or sent indirectly through the user
1306 agent.

1307 **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1308 In step 2, the responding provider (after processing the request) issues a
1309 `<ManageNameIDResponse>` message to the original requesting provider. The response may be
1310 returned directly to the requesting provider or indirectly through the user agent (if consistent with
1311 the form of the request in step 1).

1312 **4.5.3 Profile Description**

1313 In the descriptions below, the following is referred to:

Name Identifier Management Service

1315   This is the name identifier management protocol endpoint at an identity or service provider to which
1316   the `<ManageNameIDRequest>` or `<ManageNameIDResponse>` messages (or an artifact
1317   representing them) are delivered. The same or different endpoints MAY be used for requests and
1318   responses.

1319 **4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider**

1320 To initiate the profile, the requesting provider issues a `<ManageNameIDRequest>` message to another
1321 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1322 used to determine the location of this endpoint and the bindings supported by the responding provider.

Synchronous Bindings (Back-Channel)

1324   The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1325   send the request directly to the other provider. The requester MUST authenticate itself to the other
1326   provider, either by signing the `<ManageNameIDRequest>` or using any other binding-supported
1327   mechanism.

Asynchronous Bindings (Front-Channel)

1329   Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1330   asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1331   request to the other provider through the user agent.

1332   If the HTTP Redirect or POST binding is used, then the `<ManageNameIDRequest>` message is
1333   delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1334   profile defined in Section 55 is used by the other provider, which makes a callback to the requesting
1335   provider to retrieve the `<ManageNameIDRequest>` message, using for example the SOAP binding.

1336   It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1337   TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1338   `<ManageNameIDRequest>` message MUST be signed if the HTTP POST or Redirect binding is
1339   used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1340   request issuer when the artifact is dereferenced.

1341   Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1342   associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1343   little information as possible in the RelayState value unless the use of the profile does not require such
1344   privacy measures.

1345 Profile-specific rules for the contents of the `<ManageNameIDRequest>` message are included in Section
1346 4.5.4.1.

1347 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service**
1348   **Provider**

1349 The recipient MUST process the `<ManageNameIDRequest>` message as defined in [SAMLCore]. After
1350 processing the message or upon encountering an error, the recipient MUST issue a
1351 `<ManageNameIDResponse>` message containing an appropriate status code to the requesting provider
1352 to complete the SAML protocol exchange.

Synchronous Bindings (Back-Channel)

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

1357   using any other binding-supported mechanism.

1358   **Asynchronous Bindings (Front-Channel)**

1359   If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1360   Artifact bindings [SAMLBind], then the `<ManageNameIDResponse>` (or artifact) is returned through
1361   the user agent to the requesting provider's name identifier management service response endpoint.
1362   Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings
1363   supported by the requesting provider.

1364   If the HTTP Redirect or POST binding is used, then the `<ManageNameIDResponse>` message is
1365   delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1366   Resolution profile defined in Section 55 is used by the requesting provider, which makes a callback to
1367   the responding provider to retrieve the `<ManageNameIDResponse>` message, using for example the
1368   SOAP binding.

1369   It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1370   TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1371   `<ManageNameIDResponse>` message MUST be signed if the HTTP POST or Redirect binding is
1372   used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1373   response issuer when the artifact is dereferenced.

1374   Profile-specific rules for the contents of the `<ManageNameIDResponse>` message are included in
1375   Section 4.5.4.2.

4.5.4 Use of Name Identifier Management Protocol

4.5.4.1 <ManageNameIDRequest> Usage

1378   The `<Issuer>` element MUST be present and MUST contain the unique identifier of the requesting entity;
1379   the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1380   `format:entity`.

1381   The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1382   the message or using a binding-specific mechanism.

4.5.4.2 <ManageNameIDResponse> Usage

1384   The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1385   entity; the `Format` attribute MUST be omitted or have a value of
1386   `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1387   The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1388   the message or using a binding-specific mechanism.

4.5.5 Use of Metadata

1390   [SAMLMeta] defines an endpoint element, `<md:ManageNameIDService>`, to describe supported
1391   bindings and location(s) to which an entity may send requests and responses using this profile.

1392   A requester, if encrypting the principal's identifier, can use the responder's `<md:KeyDescriptor>`
1393   element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1394   settings to use, along with a public key to use in delivering a bulk encryption key.

5 Artifact Resolution Profile

1395

1396 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1397 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1398 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1399 binding, such as the SOAP binding defined in [SAMLBind].

5.1 Required Information

1400

1401 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1402 **Contact information:** security-services-comment@lists.oasis-open.org

1403 **Description:** Given below.

1404 **Updates:** None

5.2 Profile Overview

1405

1406 The message exchange and basic processing rules that govern this profile are largely defined by Section
1407 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1408 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1409 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1410 Figure 5 illustrates the basic template for the artifact resolution profile.
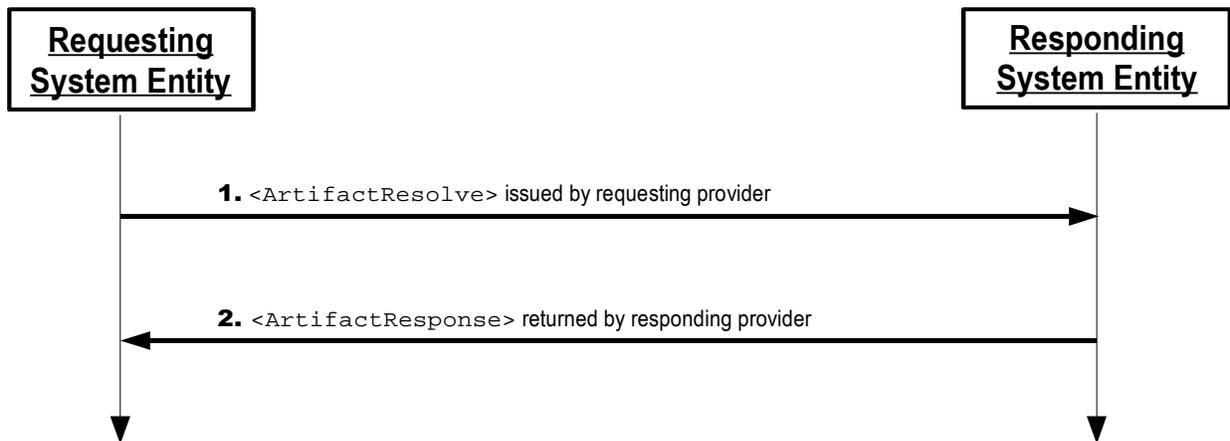

Figure 5

1411 The following steps are described by the profile.

1412 **1. <ArtifactResolve> issued by Requesting Entity**

1413 In step 1, a requester initiates the profile by sending an `<ArtifactResolve>` message to an
1414 artifact issuer.

1415 **2. <ArtifactResponse> issued by Responding Entity**

1416 In step 2, the responder (after processing the request) issues an `<ArtifactResponse>`
1417 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.

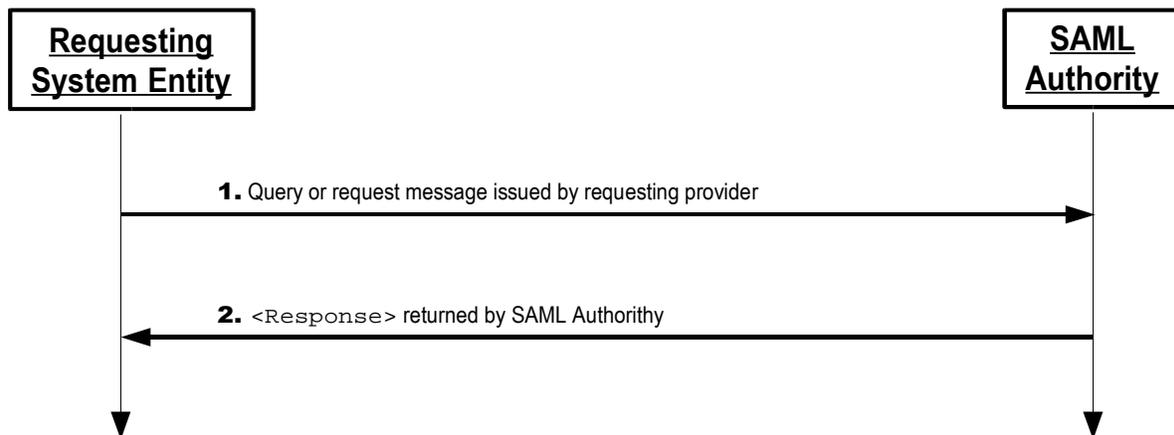Figure 6 illustrates the basic template for the query/request profile.

Figure 6

The following steps are described by the profile.

1. Query/Request issued by Requesting Entity

In step 1, a 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 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 Requesting Entity

To initiate the profile, a 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 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 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.

6.5 Use of Metadata

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

1525 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1526 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an
1527 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1528 encryption key.

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 `<NameIDMappingResponse>` message to the requester.

7.3 Profile Description

In the descriptions below, the following is referred to:

Name Identifier Mapping Service

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

7.3.1 <NameIDMappingRequest> issued by Requesting Entity

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

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

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

7.3.2 <NameIDMappingResponse> issued by Identity Provider

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

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

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

7.4 Use of Name Identifier Mapping Protocol

7.4.1 <NameIDMappingRequest> Usage

The `<Issuer>` element MUST be present.

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

7.4.2 <NameIDMappingResponse> Usage

The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding identity provider; 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.

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

7.4.2.1 Limiting Use of Mapped Identifier

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

7.5 Use of Metadata

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

The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity'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.

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 3.3 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

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

1639 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of
1640 these directory attribute types in the context of SAML attribute statements, without having to manually
1641 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.

1642 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such
1643 attributes when expressed as SAML attributes.

8.2.1 Required Information

1645 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace
1646 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

1647 **Contact information:** security-services-comment@lists.oasis-open.org

1648 **Description:** Given below.

1649 **Updates:** None.

8.2.2 SAML Attribute Naming

1651 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1652 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1653 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.
1654 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory
1655 attribute type.
1656 Example:

1657   urn:oid:2.5.4.3

1658 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,
1659 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1660 For purposes of human readability, there may also be a requirement for some applications to carry an
1661 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1662 [SAMLCore]) MAY be used for this purpose. If the definition of the directory attribute type includes one or
1663 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, SHOULD be
1664 one of the defined descriptors.

8.2.2.1 Attribute Name Comparison

1666 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1667 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the
1668 comparison.

8.2.3 Profile-Specific XML Attributes

1670 No additional XML attributes are defined for use with the `<Attribute>` element.

8.2.4 SAML Attribute Values

1672 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute
1673 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax
1674 which specifies how attribute or assertion values conforming to the syntax are to be represented when
1675 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding
1676 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of
1677 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to
1678 this profile may define attribute value formats for directory attributes whose syntaxes specify other
1679 encodings.

1680 To represent the encoding rules in use for a particular attribute value, the `<AttributeValue>` element
1681 MUST contain an XML attribute named `Encoding` defined in the XML namespace

1682   `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`.

1683   For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8
1684   character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the
1685   content of the `<AttributeValue>` element, with no additional whitespace. In such cases, the
1686   `xsi:type` XML attribute MUST be set to **xs:string**. The profile-specific `Encoding` XML attribute is
1687   provided, with a value of `LDAP`.

1688   A list of some LDAP attribute syntaxes to which this applies is:

1689 Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1690 Bit String	1.3.6.1.4.1.1466.115.121.1.6
1691 Boolean	1.3.6.1.4.1.1466.115.121.1.7
1692 Country String	1.3.6.1.4.1.1466.115.121.1.11
1693 DN	1.3.6.1.4.1.1466.115.121.1.12
1694 Directory String	1.3.6.1.4.1.1466.115.121.1.15
1695 Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1696 Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1697 IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1698 INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1699 LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1700 Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1701 Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1702 Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1703 Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1704 Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1705 Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1706 Octet String	1.3.6.1.4.1.1466.115.121.1.40
1707 OID	1.3.6.1.4.1.1466.115.121.1.38
1708 Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1709 Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1710 Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1711 Printable String	1.3.6.1.4.1.1466.115.121.1.44
1712 Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1713 Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1714 UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1715   For all other LDAP syntaxes, the attribute value is encoded, as the content of the `<AttributeValue>`
1716   element, by base64-encoding [RFC2045] the encompassing ASN.1 OCTET STRING-encoded LDAP
1717   attribute value. The `xsi:type` XML attribute MUST be set to **xs:base64Binary**. The profile-specific
1718   `Encoding` XML attribute is provided, with a value of "`LDAP`".

1719   When comparing SAML attribute values for equality, the matching rules specified for the corresponding
1720   directory attribute type MUST be observed (case sensitivity, for example).

1721 8.2.5 Profile-Specific Schema

1722   The following schema defines the profile-specific `Encoding` XML attribute:

```
1723    <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1724          xmlns="http://www.w3.org/2001/XMLSchema"
1725          version="2.0">
1726          <attribute name="Encoding" type="string"/>
1727    </schema>
```

1728 8.2.6 Example

1729   The following is an example of a mapping of the "givenName" directory attribute, representing the SAML
1730   assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory
1731   String.

```
1732    <saml:Attribute xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1733                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1734                    Name="urn:oid:2.5.4.42" FriendlyName="givenName">
1735            <saml:AttributeValue xsi:type="xs:string"
1736                    x500:Encoding="LDAP">Steven</saml:AttributeValue>
1737    </saml:Attribute>
```

8.3 UUID Attribute Profile

1739 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1740 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with
1741 a UUID.

8.3.1 Required Information

1743 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1744 **Contact information:** security-services-comment@lists.oasis-open.org

1745 **Description:** Given below.

1746 **Updates:** None.

8.3.2 UUID and GUID Background

1748 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1749 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs
1750 were originally used in the Network Computing System (NCS), and then used in the Open Software
1751 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1752 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1753 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1754 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1755 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

```
1756    f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

1757 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1758 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

8.3.3 SAML Attribute Naming

1760 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1761 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1762 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace
1763 described in [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt] is used. In this approach the
1764 `Name` XML attribute is based on the URN form of the underlying UUID that identifies the attribute.

1765 Example:
```
1766    urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

1767 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1768 in the `Name` XML attribute.

1769 For purposes of human readability, there may also be a requirement for some applications to carry an
1770 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1771 [SAMLCore]) MAY be used for this purpose.

8.3.3.1 Attribute Name Comparison

1773 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute

1774 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1775 `FriendlyName` attribute plays no role in the comparison.

8.3.4 Profile-Specific XML Attributes

1777 No additional XML attributes are defined for use with the `<Attribute>` element.

8.3.5 SAML Attribute Values

1779 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1780 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST
1781 be set to **xs:anyURI**.

1782 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`
1783 element.

8.3.6 Example

1785 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a
1786 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

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

8.4 DCE PAC Attribute Profile

1793 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
1794 and values. It is used to standardize a mapping between the primary information that makes up a DCE
1795 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
1796 Section 8.3.

8.4.1 Required Information

1798 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
1799 assigned in the corresponding DCE PAC attribute profile schema document [SAMLDCE-xsd])

1800 **Contact information:** security-services-comment@lists.oasis-open.org

1801 **Description:** Given below.

1802 **Updates:** None.

8.4.2 PAC Description

1804 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
1805 information is common across principals and makes up the bulk of a DCE identity:
1806 • The principal's DCE "realm" or "cell"

1807 • The principal's unique identifier

1808 • The principal's primary DCE local group membership

1809 • The principal's set of DCE local group memberships (multi-valued)

1810 • The principal's set of DCE foreign group memberships (multi-valued)

1811 The primary value(s) of each of these attributes is a UUID.

8.4.3 SAML Attribute Naming

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

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

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

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

8.4.3.1 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 [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. 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 defines the profile-specific XML attributes and a complex type used in an `xsi:type` specification:

```
<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"
        version="2.0">
        <attribute name="Realm" type="anyURI"/>
        <attribute name="FriendlyName" type="string"/>
        <complexType name="DCEValueType">
                <simpleContent>
                        <extension base="anyURI">
                                <attribute ref="dce:Realm" use="optional"/>
                                <attribute ref="dce:FriendlyName" use="optional"/>
                        </extension>
                </simpleContent>
        </complexType>
</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`.

1858 Note that such use of `xsi:type` will require validating attribute consumers to include the extension
1859 schema defined by this profile.

8.4.6.1 Realm

1861 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

1862 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

1863 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
1864 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
1865 realm's string name.

8.4.6.2 Principal

1867 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

1868 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

1869 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
1870 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
1871 the principal's string name.

1872 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1873 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1874 8.4.6.1).

8.4.6.3 Primary Group

1876 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

1877 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

1878 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
1879 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
1880 the group's string name.

1881 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1882 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1883 8.4.6.1).

8.4.6.4 Groups

1885 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

1886 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

1887 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE group membership
1888 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
1889 the group's string name.

1890 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1891 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1892 8.4.6.1).

8.4.6.5 Foreign Groups

1894 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

1895 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

1896 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE foreign group
1897 membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute
1898 containing the group's string name.

1899 The profile-specific `Realm` XML attribute MUST be included and MUST contain a UUID in URN form
1900 identifying the DCE realm/cell of the foreign group.

8.4.7 Example

1902 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE
1903 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local
1904 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">
    urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b
    </saml:AttributeValue>
  </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

8.5 XACML Attribute Profile

1952 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
1953 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
1954 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
1955 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
1956 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
1957 XACML attributes and used as input to XACML authorization decisions.

8.5.1 Required Information

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

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

Description: Given below.
Updates: None.

8.5.2 SAML Attribute Naming

The `NameFormat` XML attribute in `<Attribute>` elements MUST be
`urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.
The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

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, but is not translatable into the XACML attribute equivalent.

8.5.2.1 Attribute Name Comparison

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

8.5.3 Profile-Specific XML Attributes

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

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

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

8.5.4 SAML Attribute Values

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

8.5.5 Profile-Specific Schema

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

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

8.5.6 Example

1997 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
1998 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
1999 attribute profiles when they are compatible with each other.

```
<saml:Attribute
xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
        xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
                xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
                ldapprof:Encoding="LDAP"
                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">By-Tor</saml:AttributeValue>
</saml:Attribute>
```

9 References

[AES]   FIPS-197, Advanced Encryption Standard (AES), available from http://www.nist.gov/.

[Anders]   A suggestion on how to implement SAML browser bindings without using "Artifacts", http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt.

[ASN.1]   Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation, ITU-T Recommendation X.680, July 2002. See http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680.

[eduPerson]   eduPerson.ldif. See http://www.educase.edu/eduperson.

[LDAP]   J. Hodges et al., Lightweight Directory Access Protocol (v3): Technical Specification, IETF RFC 3377, September 2002. See http://www.ietf.org/rfc/rfc3377.txt.

[Mealling]   P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03. January 2004

[MSURL]   Microsoft technical support article, http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP.

[NSCookie]   Persistent Client State HTTP Cookies, Netscape documentation. See http://wp.netscape.com/newsref/std/cookie_spec.html.

[Rescorla-Sec]   E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations, http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt.

[RFC1738]   Uniform Resource Locators (URL), http://www.ietf.org/rfc/rfc1738.txt

[RFC1750]   Randomness Recommendations for Security. http://www.ietf.org/rfc/rfc1750.txt

[RFC1945]   Hypertext Transfer Protocol -- HTTP/1.0, http://www.ietf.org/rfc/rfc1945.txt.

[RFC2045]   Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, http://www.ietf.org/rfc/rfc2045.txt

[RFC2119]   S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt.

[RFC2246]   The TLS Protocol Version 1.0, http://www.ietf.org/rfc/rfc2246.txt.

[RFC2256]   M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3, December 1997

[RFC2279]   UTF-8, a transformation format of ISO 10646, http://www.ietf.org/rfc/rfc2279.txt.

[RFC2616]   Hypertext Transfer Protocol -- HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt.

[RFC2617]   HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617, http://www.ietf.org/rfc/rfc2617.txt.

[RFC2798]   M. Smith, Definition of the inetOrgPerson LDAP Object Class, IETF RFC 2798, April 200. See http://www.ietf.org/rfc/rfc2798.txt.

[RFC2965]   D. Cristol et al., HTTP State Management Mechanism, IETF RFC 2965, October 2000. See http://www.ietf.org/rfc/rfc2965.txt.

[RFC3061]   M. Mealling, A URN Namespace of Object Identifiers, IETF RFC 3061, February 2001. See http://www.ietf.org/rfc/rfc3061.txt.

[SAMLBind]   S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-2.0-cd-02. See http://www.oasis-open.org/committees/security/.

[SAMLCore]   S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-core-2.0-cd-02. See http://www.oasis-open.org/committees/security/.

[SAMLDCE-xsd]	S. Cantor et al., SAML DCE PAC attribute profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-dce-2.0. See http://www.oasis-open.org/committees/security/.
[SAMLECP-xsd]	S. Cantor et al., SAML ECP profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-ecp-2.0. See http://www.oasis-open.org/committees/security/.
[SAMLGloss]	J. Hodges et al.*, Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-glossary-2.0-cd-02. See http://www.oasis-open.org/committees/security/.
[SAMLX500-xsd]	S. Cantor et al., SAML LDAP attribute profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-ldap-2.0. See http://www.oasis-open.org/committees/security/.
[SAMLMeta]	S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-metadata-2.0-cd-02. See http://www.oasis-open.org/committees/security/.
[SAMLReqs]	Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002, http://www.oasis-open.org/committees/security/.
[SAMLSec]	F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-sec-consider-2.0-cd-02. See http://www.oasis-open.org/committees/security/.
[SAMLWeb]	OASIS Security Services Technical Committee website, http://www.oasis-open.org/committees/security.
[SAMLXAC-xsd]	S. Cantor et al., SAML XACML attribute profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-xacml-2.0. See http://www.oasis-open.org/committees/security/.
[Schema1]	H. S. Thompson et al. *XML Schema Part 1: Structures.* World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/. Note that this specification normatively references [Schema2], listed below.
[Schema2]	Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001, http://www.w3.org/TR/xmlschema-2/
[SESSION]	RL "Bob" Morgan, Support of target web server sessions in Shibboleth, http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt
[ShibMarlena]	Marlena Erdos, Shibboleth Architecture DRAFT v1.1, http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html .
[SOAP1.1]	D. Box et al., Simple Object Access Protocol (SOAP) 1.1, World Wide Web Consortium Note, May 2000, http://www.w3.org/TR/SOAP.
[SSL3]	A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November 1996.
[WEBSSO]	RL "Bob" Morgan, Interactions between Shibboleth and local-site web sign-on services, http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt
[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.
[XMLEnc]	D. Eastlake et al., XML Encryption Syntax and Processing, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/, World Wide Web Consortium.
[XMLSig]	D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web Consortium, http://www.w3.org/TR/xmldsig-core/.
[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

2105 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

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
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Tim Alsop, CyberSafe Limited
- Paul Madsen, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Paula Austel, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Prateek Mishra, Netegrity
- Forest Yin, Netegrity
- Peter Davis, Neustar
- Frederick Hirsch, Nokia
- John Kemp, Nokia
- Senthil Sengodan, Nokia
- Scott Kiester, Novell
- Cameron Morris, Novell
- Charles Knouse, Oblix
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Jeff Hodges, Sun Microsystems
- Eve Maler, Sun Microsystems

2148 • Ronald Monzillo, Sun Microsystems

2149 • Emily Xu, Sun Microsystems

2150 • Mike Beach, Boeing

2151 • Greg Whitehead, Trustgenix

•

2152 The editors also would like to acknowledge the following people for their contributions to previous versions
2153 of the OASIS Security Assertions Markup Language Standard:

2154 • Stephen Farrell, Baltimore Technologies

2155 • David Orchard, BEA Systems

2156 • Krishna Sankar, Cisco Systems

2157 • Zahid Ahmed, CommerceOne

2158 • Carlisle Adams, Entrust

2159 • Tim Moses, Entrust

2160 • Nigel Edwards, Hewlett-Packard

2161 • Joe Pato, Hewlett-Packard

2162 • Bob Blakley, IBM

2163 • Marlena Erdos, IBM

2164 • Marc Chanliau, Netegrity

2165 • Chris McLaren, Netegrity

2166 • Lynne Rosenthal, NIST

2167 • Mark Skall, NIST

2168 • Simon Godik, Overxeer

2169 • Charles Norwood, SAIC

2170 • Evan Prodromou, Securant

2171 • Robert Griffin, RSA Security (former editor)

2172 • Sai Allarvarpu, Sun Microsystems

2173 • Chris Ferris, Sun Microsystems

2174 • Emily Xu, Sun Microsystems

2175 • Mike Myers, Traceroute Security

2176 • Phillip Hallam-Baker, VeriSign (former editor)

2177 • James Vanderbeek, Vodafone

2178 • Mark O'Neill, Vordel

2179 • Tony Palmer, Vordel

2180 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2181 input to the OASIS Security Assertions Markup Language specifications:

2182 • Thomas Gross, IBM

2183 • Birgit Pfitzmann, IBM

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 2004. *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.